Gazeta de informatică
Revista viitorilor specialiști în tehnologia informației
Ianuarie 2004
Vol. 14/1
Lei 30.000

din sumarul numărului:

actual

Plus

concurs
Soluțiile problemelor - Clasament

probleme
Concursul de informatică "N. Păun"

internet
Interogarea bazelor de date MySQL

...itate (bucăți)

TALON DE COMANDĂ

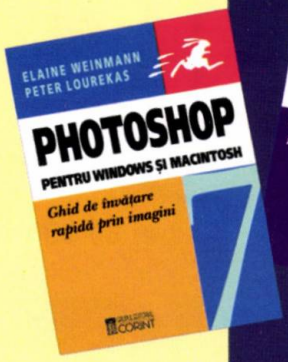
Corespondență Răspuns
CR
Se taxează la destinație

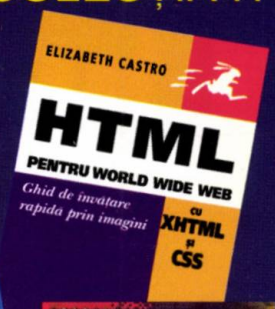
GRUPUL EDITORIAL CORINT

Ultimele apariții în COLECȚIA IT

- ☐ **INIȚIERE ÎN DESIGN** de Robin Williams. Preț: 125 000* lei
- ☐ **MICROSOFT OFFICE XP PENTRU WINDOWS GHID DE ÎNVĂȚARE RAPIDĂ PRIN IMAGINI** de Steve Sagman. Preț: 249 000* lei
- ☐ **DESIGN PENTRU WEB** de Robin Williams, John Tollet. Preț: 299 000* lei
- ☐ **INIȚIERE ÎN WINDOWS XP GHID DE PREZENTARE A SISTEMULUI DE OPERARE** de Larry Magid. Preț: 225 000* lei
- ☐ **QUARKXPRESS 4 și 5 SHORTCUTS PENTRU PC ȘI MACINTOSH** de David Blatner. Preț: 120 000* lei
- ☐ **CALCULATOARE ȘI INTERNET. DICȚIONAR EXPLICATIV** de Random House Webster's. Preț: 140 000* lei
- ☐ **PHOTOSHOP 7 PENTRU WINDOWS ȘI MACINTOSH GHID DE ÎNVĂȚARE RAPIDĂ PRIN IMAGINI** de Elaine Weinmann, Peter Lourekas. Preț: 350 000* lei
- ☐ **HTML PENTRU WORLD WIDE WEB GHID DE ÎNVĂȚARE RAPIDĂ PRIN IMAGINI** de Elizabeth Castro. Preț: 375 000* lei
- ☐ **MACROMEDIA® DREAMWEAVER® MX ÎN 24 DE LECȚII** de Betsy Bruce. Preț: 325 000* lei

* fară TVA





EXPEDITOR:
Nume: ..... Prenume: .....
Str. ..... Nr. ..... Bl. ..... Sc. ..... Et. ..... Ap. .....
Localitate ..... Cod poștal ..... Jud. / Sect. ..... Tel. .....

Completați talonul de comandă conform opțiunilor dumneavoastră și introduceți-l – netimbrat – în orice cutie poștală.

DESTINATAR:
Editura CORINT,
CP 1-591,
București
tel.: 223.19.28
222.19.49
tel/fax: 222.40.34

Cine are carte, are parte!

... de înțelegerea rețelelor!

Cea mai bună introducere din lume în rețele de calculatoare - complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o reducere semnificativă!

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65; 0721-927.499

Vol. 14, Nr. 1 /2004

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Director Bursele Agora:
Mihai Scorțaru
Redactor-șef:
Claudiu Soroiu
Redactor-șef adjunct:
Carmen Șușman
Redactori:
prof. *Eugen Ionescu*
Mihai Stroe

Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
400091, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/592.422, 0265/266.516
Fax: 0265/266.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359

Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu
Denis Marioara
Marius Comărniceanu
Difuzare:
Florin Dinu
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134

Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

sumar

actual.. PAG. 3

IMAGINE CUP
Alexandru PRUTEANU
~ concurs Microsoft ~

Pledoarie pentru JAVA
Mihai SCORȚARU
~ răspuns la ~
~ replicile cititorilor ~

probleme.. PAG. 8

Concursul de Informatică "NICOLAE PĂUN"

Concursul de Informatică "ȘTEFAN ODOBLEJA"

~ enunțurile celor 15 ~
~ probleme propuse ~
~ spre rezolvare ~

soluții.. PAG. 17

Olimpiada BALCANICĂ de informatică
Mihai STROE

Concursul național de programare "STELELE INFORMATICII"
Mihai STROE

~ soluțiile problemelor ~
~ propuse spre rezolvare ~
~ la cele două concursuri ~

concurs.. PAG. 27

Despre rundele #01-#05
~ soluțiile problemelor ~

Clasament
~ Top 50 ~

Rundele #06-#15
~ enunțurile problemelor ~
~ propuse spre rezolvare ~

mate.. PAG. 35

ALGORITMI - CALCULABILITATE
Horia GEORGESCU
~ ultima parte ~
~ alte metode de ~
~ calculabilitate ~

Șiruri RECURENTE
Eugen NODEA
~ rezolvare recurențelor ~
~ și recurențe celebre ~

internet.. PAG. 44

Pagini WEB cu PHP 4
Mihai SCORȚARU
Claudiu SOROIU
~ prelucrarea imaginilor ~
~ interogarea bazelor ~
~ de date MySQL ~

Verba volant, scripta manent...

Uităm să ne exprimăm sau "ȘI AȘA E BINE"?

Cu toții suntem *cititori*, iar unii mai și *scriem* din când în când ceva. Elevii scriu teze și lucrări de control, studenții scriu lucrări de examen, profesorii scriu manuale sau cursuri. Unii dintre ei mai formulează și enunțuri de probleme pentru diferite ocazii: concursuri, examene etc.

Mie mi-a fost scris în cartea vieții să citesc manuale de informatică în "devenire"..., să citesc multe enunțuri de probleme, propuse spre rezolvare la diverse concursuri de informatică și să mai citesc și lucrări scrise de elevi și studenți cu diferite ocazii. Le citesc și sunt tot mai supărată. Indiferent în ce limbă este scris textul cu pricina, întâlnesc foarte multe propoziții fără sens, fără predicat sau fără subiect, și multe cuvinte nepotrivite. (Greșelile de ortografie nici nu mai vreau să le amintesc.) Revenind la conținutul acestora, (manuale, enunțuri de probleme): încerc să ghicesc ce a vrut să spună autorul lor... Câteodată se poate descifra o asemenea frază, mai ales dacă este vorba despre un manual, respectiv atunci când știu ce ar trebui să fie scris sub anumite titluri. Dar, se întâmplă frecvent că doar sper că am descifrat câte un astfel de text.

Mă gândesc întristată la cei "nevinovați" care vor înțelege altceva, sau dimpotrivă, nu vor înțelege nimic. Atunci când cineva formulează, de exemplu, un enunț de problemă, se știe că *în cap* are deja tot: ce se dă și ce se cere, are și rezolvarea în care a declarat, de exemplu, "un șir de cel mult 1000 de numere întregi" etc. Dar acest lucru nu scrie în enunț. Nu scrie clar nici cerința propriu-zisă. Se vorbește despre un *subșir*, dar analizând exemplul dat în enunț, se observă că rezultatul este *subsecvență*. Oare unde este bine? Acolo, unde scrie *subșir*? (deci exemplul nu este corespunzător ales), sau exemplul este corect și rezultă că în enunț, de fapt, este vorba de *subsecvență*.

Dacă este vorba de un concurs, problema neclarităților, de regulă, se rezolvă în mod fericit. Concurentul pune o întrebare, și dacă a fost învățat s-o pună corect, primește un răspuns, *posibil* corect. Cum adică "posibil", veți întreba! Condiția ca acel răspuns monosilabic "DA" respectiv "NU" să fie corect este ca cel care răspunde - să fie *autorul* problemei, să înțeleagă limba în care s-a formulat întrebarea și să nu fie atât de obosit încât să înțeleagă altceva decât s-a formulat în întrebare. Dar, de multe ori nici întrebarea nu este "inteligentă". Și atunci, răspunsul va fi "Fără comentarii", după cum bine se știe. Cine să-l învețe pe concurent cum să-și formuleze întrebarea? Profesorul, veți răspunde. Da, dar profesorul, de multe ori, nu vorbește nici el suficient de clar. Știu, știu, sunt excepții. Sau, mai bine zis, cu toții sperăm că majoritatea sunt aceste excepții: cei care știu informatică, se exprimă clar și scriu corect.

Din păcate iar vine realitatea să dovedească contrariul. Cine credeți că scriu manuale de informatică, cine credeți că formulează enunțuri de probleme pentru concursuri? Cei mai buni dintre profesori, nu? Dar atunci cum se explică faptul că unii dintre ei scriu manuale în care nici un subprogram nu are parametri sau atunci când de fapt vor să zică "*folosim variabila de tip întreg x*" scriu "*folosim tipul întreg x*" sau, așa cum am citit recent într-un manual: "*constantele se identifică printr-un nume*" și vă asigur că niciunde nu scria că este vorba despre constante simbolice. Dacă așa scrie în manual, așa învață elevul - pentru acest răspuns primește 10 - și nimeni nu îl poate consola, dar nici lămuri când la admitere pentru un astfel de răspuns nu primește puncte. Aș putea să dau mii de asemenea exemple, dar voi coborî un "nivel" în argumentația mea și voi lăsa la o parte conținutul științific și voi reveni la deficiențele de exprimare în limbile noastre materne.

Adolescenții de azi nu mai acordă importanță modului în care se exprimă atunci când vorbesc și atunci când scriu. Motivele sunt multiple: televizorul, jocurile etc., dar voi adăuga că din păcate și școala contribuie. Dacă profesorul de chimie scrie cu greșeli de ortografie schița pe tablă, dacă profesorul de geografie nu corectează greșelile de exprimare, dacă profesorul de informatică nu le cere elevilor să vorbească în propoziții întregi...

Oare se apropie un viitor în care lumea va accepta texte, indiferent de corectitudinea lor gramaticală? Cum se vor descifra aceste texte? Sau ne vom opri la un nivel de superficialitate la care "încă se poate înțelege", în rest, vom lua în brațe "las' că-i bine și așa"?

Sper că nu ne îndreptăm spre o astfel de lume..., așa că vă spun, "*nu, nu e bine așa*"!

Clara Ionescu

Imagine Cup

Alexandru Pruteanu

Imagine Cup este o competiție internațională proiectată pentru a inspira și evidenția excelența în inovația tehnică. Prin Imagine Cup, Microsoft încurajează exprimarea ideilor studenților și premiază cele mai bune soluții, oferind șansa afirmării în întreaga lume!

Imagine Cup este la a doua ediție mondială și la prima desfășurare în *România*. Considerăm că este o nouă oportunitate pentru studenții noștri de a dovedi încă o dată potențialul extraordinar de care dispune școala românească.

Secțiuni

În acest an competiția are patru secțiuni. Încercăm în acest mod să surprindem cât mai multe arii de interes, astfel încât un număr cât mai mare de studenți, cu preocupări dintre cele mai diverse, să participe la această competiție. Pentru fiecare secțiune există un proces eliminatoriu de evaluare, în urma căruia candidații admiși vor fi invitați să participe la faza finală. Din acest motiv, secțiunile se numesc și invitații.

Invitațiile din acest an sunt: *renderizare*, *film de scurt metraj*, *algoritmică* și *inginerie software*. Primele trei secțiuni se vor desfășura *online*, iar secțiunea dedicată *ingineriei software* se va desfășura local (faza întâi se va susține în *România* și va fi deschisă pentru studenții din *România* și *Republica Moldova*), regional (faza a doua va avea loc pentru *Europa de Est* și va fi organizată în *Cambridge*, *Anglia*) și global (finala va fi organizată în *Sao Paolo*, *Brazilia*).

Vor fi acordate premii în bani, *software* și echipamente *hardware*.

Renderizare

Grafica generată pe calculator este un domeniu extrem de dinamic și are un impact imens asupra felului în care oamenii percep și interacționează cu realitatea.

Invitația se adresează pasionaților de grafică generată pe calculator - *Imagine Cup* pune la dispoziție un cadru în care talentul și îndemânarea artiștilor să fie apreciate și recunoscute.

Instrumentele de bază în încercarea de a dovedi că sunteți un artist extraordinar trebuie să fie *Visual Studio .NET* și *DirectX* 9.0.

Echipele participante trebuie să aibă până la patru membri.

Film de scurt metraj

Sunteți invitați să participați la această secțiune pentru a spune o poveste folosind tehnologia.

Tema acestei secțiuni este "*cultura inovației*". Cum percepeți voi această cultură? Dând un răspuns la această întrebare veți avea ocazia să îi sensibilizați pe cei din jur cu perspectiva voastră, cu un documentar de 5 minute sau cu un clip de câteva zeci de secunde.

Clipul poate fi realizat folosind orice instrument, important este ca rezultatul să poată fi vizualizat cu *Windows Media Player* (se recomandă folosirea unuia dintre formatele *avi* sau *mpeg*).

O echipă participantă nu trebuie să numere mai mult de patru membri.

Algoritmică

Pasionații de informatică și cei care s-au antrenat să rezolve probleme dificile de informatică în timp real au la dispoziție această secțiune. Acesta este un concurs *online* de programare în cadrul căruia aveți la dispoziție un limbaj compatibil cu platforma *.NET* (exemplu *C#*) pentru a implementa algoritmii pentru problemele propuse.

Acesta este un concurs individual la care contează capacitatea de a implementa rapid și corect algoritmii.

Inginerie software

Tema secțiunii dedicate ingineriei *software* a competiției *Imagine Cup* 2004 este "*Să ne imaginăm o lume în care tehnologiile inteligente simplifică fiecare zi din viața noastră...*".

Dacă aveți o idee strălucită pentru un produs care să se încadreze în tematica acestui concurs, atunci aveți șansa de a o pune în valoare. În acest an se pune un accent deosebit pe soluțiile care utilizează tehnologii inteligente și care conectează diferite tipuri de dispozitive mobile pe platforma *.NET*.

Pe *site*-ul *Microsoft România*, pe pagina **www.microsoft.com/Romania/Educatie/Studenti/** puteți afla mai multe informații legate de premiile oferite și de regulamentul de participare. Eventualele întrebări pot fi puse la adresa **roacad@microsoft.com** (*Microsoft România - Academic Program*).

Pentru a afla mai multe detalii despre cele patru secțiuni puse la dispoziție sunteți invitați să vizitați *site*-ul oficial al competiției *Imagine Cup* - **www. imaginecup.com.**

Pledoarie pentru JAVA

Mihai Scorțaru

Ne-am bucurat foarte mult când am văzut mulțimea de mesaje pe care le-am primit, referitor la articolul din numărul trecut. Așa cum am promis, vom permite tuturor celor care doresc să își exprime părerea referitoare la acest subiect. Sperăm că această discuție va continua și în numerele următoare.

Laviniu Aurelian Bădulescu
profesor, Colegiul Național "Elena Cuza", Craiova

Deși sunt - încă - în al patrulea deceniu de viață, am parcurs - învățând mai mult singur - aproape toată istoria calculatoarelor în România, de la cartelele Felix-ului, prin benzile minicalculatoarelor Coral și Independent, prin "macro" dischetele microcalculatoarelor Junior și TPD, la generațiile succesive de PC-uri compatibile IBM: XT, AT 286, 386 ... până la Pentium 4.

De ceva timp a apărut Java. Nu am nimic împotriva limbajului Java. Dar de ce să renunțăm la algoritmică? Eu cred că limbajele trec și algoritmii rămân. Sunt de acord să se predea Java pe trunchiul comun la clasele de matematică informatică (intensiv sau nu). Pentru că trebuie să ținem pasul cu tendințele din domeniu de pe plan mondial. Și pentru asta trebuie să învățăm permanent. Consider că este vital - în primul rând pentru mine - să învăț. Și apoi, "Docendo discitur".

Dar ... profesorul de informatică nu are nici un statut "privilegiat" în raport cu care ar trebui să învețe și să muncească mai mult decât ceilalți profesori. Profesorii de informatică nu sunt mai bine văzuți decât cei de matematică (cu care s-ar putea încerca o comparație). După cele constatate de mine în cei 12 ani de când predau la clase de profil informatică în liceu, aș putea spune că într un "top al respectabilității", locul lor este undeva la coadă, puși alături de cei de educație fizică și poate de cei de religie.

Deci să învățăm!

Dar cum?

Prin C.C.D. se fac tot felul de cursuri ca "să-i învețe informatică" pe cei de alte specialități. Și vin destui. Mai ales pentru "carton". Dar de cursuri de Java pentru cei de informatică nu prea am auzit.

Poate vom găsi cursuri în altă parte!

Un master (2 ani) pe Inginerie software - de altfel, o raritate în peisajul postuniversitar românesc - fusese inițial (septembrie 2003) anunțat la 400 de euro pe an (adică vreo 17 000 000 lei). Ori un profesor de informatică de liceu nu ajunge - după aproape 20 de ani vechime, cu gradul I - la un salariu de încadrare de 5000000 lei pe lună. Se dau subvenții consistente pentru căldură, dar pentru a ține "caldă" mintea unui profesor de informatică, nimic. S-a investit enorm (la capitolul informatică) în "ultimele dotări, în școli", dar cred că sumele acestea mai bine gestionate ajungeau și pentru plata unor cursuri pentru profesorii de informatică.

"Oricum, faptul că ești prost plătit nu reprezintă o scuză pentru a nu-ți face meseria." Nu o scuză, dar un impediment. Și unul mare de tot! Știm foarte bine cât de scumpe sunt cărțile de informatică. Și cât de repede se uzează moral (de aici și o altă constatare: nu se pune problema de a le găsi în biblioteci).

Poate ar trebui să căutăm materiale pe Internet. A-ți permite Internet acasă este un lux (ceva mai mult de 10% din salariu) pentru un profesor de informatică, care oricum trebuie din 3 în 2 ani să-și cumpere măcar un calculator.

Cine nu vrea să învețe!?

A învăța este în primul rând o "igienă" a creierului. Lecturând o carte despre bolile vârstei a III-a, despre degenerările gen Parkinson am întâlnit următoarea afirmație: "Folosește ca să nu pierzi!". Plutarh la 80 de ani a învățat limba greacă. Suntem oameni în primul rând prin faptul că gândim și ca să gândești trebuie tot timpul să înveți. "Memoria scade dacă nu o folosești", spune Cicero în De senectute. Cei care nu sunt de acord cu aceste principii sunt "propriii lor călăi", cum plastic își intitulează Terențiu o carte.

Dar ca să înveți îți trebuie cursuri, cărți, Internet. Și mai trebuie să-l hrănești și pe "fratele porc" cum numește trupul Constantin Noica. Și să-l hrănești bine! Efortul intelectual consumă mult mai mult decât cel fizic din resursele organismului. Ori asta costă! Din ce le plătești? Într-o emisiune la TV, anul trecut, Eugen Simion spunea că "nu se poate face educație în sărăcie".

Cu credința că se dorește un dialog "sine ira et studio", am scris aceste rânduri cu nădejdea că se poate rezolva ceva prin cei care au dragoste adevărată pentru profesia de dascăl. Iar revista GInfo am constatat de a lungul anilor că și-a identificat interesele cu binele învățământului de informatică românesc.

Și acum, răspunsul...

Domnule profesor, în primul rând permiteți-mi să vă felicit pentru tot ceea ce încercați să faceți pentru a vă face meseria. Din nefericire, sunteți printre puținii profesori de informatică din România care consideră că este de datoria dascălului să învețe permanent.

Nu trebuie să uităm niciodată că sistemul de învățământ este destinat, în primul rând elevilor, și abia apoi profesorilor. În primul rând, trebuie ca elevul să găsească o utilitate în a veni la școală și pentru aceasta este datoria profesorului să îl învețe pe elev, ceea ce îi este de real folos celui din urmă.

Nu putem avea pretenția ca profesorul să știe "totul", dar nici nu putem accepta ca acesta să se complacă în propria sa neștiință. De fiecare dată când afirm că ceea ce se predă în liceele românești este depășit, mi se dau replici de genul: "Și ce vrei? Să îi învățăm întotdeauna ultimele tehnologii care vor fi oricum depășite atunci când ei vor ajunge să le folosească?" Acest "salt" dintr-o extremă în alta nu poate indica, în opinia mea, decât frustrarea persoanelor respective. Sunt sigur că ar dori și ei să cunoască "ultimele tehnologii", dar nu fac nimic în acest sens.

Aveți dreptate atunci când spuneți că salariile mici ale profesorilor constituie un impediment. Dar, sunt convins că, după cum își fac treaba, mai mult de jumătate dintre ei nu le merită nici măcar pe acestea. Nimeni nu poate avea pretenția unui salariu mai mare pe baza unor promisiuni de genul "dacă primesc mai mulți bani, fac și eu mai multe!". Sunt curios cât ar rezista într-o companie "care se respectă" o persoană care, pentru a învăța ceva nou, consideră că i se cuvin mai mulți bani.

A învăța lucruri noi ar trebui să fie o "sarcină de serviciu" și nu o opțiune a angajatului. Dacă credem că nu are rost să ne chinuim "pentru banii ăștia", atunci profesorii vor avea întotdeauna "banii ăștia".

Salariile profesorilor nu se pot mări. Nu atâta timp cât nivelul de trai rămâne cel de acum. Este de datoria tuturor, inclusiv a profesorilor, să contribuie la creșterea acestui nivel de trai. Critica mea s-a adresat profesorilor de informatică deoarece îmi exprim părerile în *GInfo*. Exemple similare pot fi găsite în toate celelalte domenii...

Mai afirmați faptul că există anumite cursuri, dar care au prețuri pe care profesorii nu și le pot permite. Este adevărat! Astfel de cursuri sunt pentru cei care au bani și majoritatea profesorilor nu au! Totuși, eu cred că un profesor își poate permite un astfel de curs măcar o dată la câțiva ani. În plus, ar trebui să nu își plătească cursurile, dar... asta nu se va întâmpla prea curând. Problema cu aceste cursuri este, în opinia mea, că nici măcar nu sunt foarte utile. Puține dintre ele sunt cu adevărat importante. Ați menționat un curs de "ingineria programării". Printr-o coincidență, tocmai m-am întors de la un examen la o materie numită *Tehnici avansate de ingineria programării*. Nu pot să îmi imaginez la ce îmi poate folosi ceea ce am învățat. Deși denumirea materiei pare interesantă, conținutul era, în realitate, o matematică abstractă și complet inutilă pentru 99% dintre cei care ajung să urmeze un astfel de curs în România.

În final, vă mulțumesc pentru scrisoarea pe care ne-ați trimis-o. Din motive de spațiu, nu am reușit să "dau o replică" la toate problemele pe care le-ați ridicat. Oricum, discuția va continua în numerele următoare. Probabil că vor fi și alți cititori care vor dori să vă dea o replică...

Lucian Ilea

profesor, Colegiul Național "Emil Racoviță", Cluj

Acum vreo două zile am cumpărat GInfo numărul pe luna decembrie. Am ajuns pe la ora 10 acasă, după ce restul zilei l-am petrecut căutând rezolvări la câteva probleme. Speram să mă delectez cu o "cafea". În loc de delectare am găsit o critică feroce a tot ceea ce se numește informatică și profesor de liceu. Nu este nici prima, nici ultima dată. Am citit la sfârșit că așteptați ceva contrapăreri. Una dintre ele am găsit-o pe site-ul **http://www.intraweb.ro/txt/Articole/Limbaje/Oberon/show** *recomandată de dl. Emil Onea pe ceva grup de discuții (tinfo). Articolul este scris de Mircea Sârbu :). Deci, asta ar fi referitor la limbaje de programare. Referitor la profesori nu știu dacă are rost să îi tot criticați atâta. Până acum vreo zece ani toată lumea se pricepea la fotbal și la agricultură. Acum toată lumea se pricepe la învățământ. Și eu și oricare am avut profesori de la care nu am învățat mare lucru, în special la facultate. Dar probabil că învățământul de piață la care s-a ajuns le va cere și rezultate concrete. Și mie mi se părea floare la ureche să fii profesor. Între timp mi-am schimbat părerea. Principala dificultate nu este aceea de a învăța Java, sau C#, sau Perl, sau Python, sau Ritzy-Pitzy. Nimeni nu le cere profesorilor de fizică să obțină energie nucleară la clasă sau celor de atelier să construiască zgârie-nori. În schimb, celor de informatică li se cere în permanență să cunoască nu știu ce tehnologie nou apărută sau reluată. Nu are nimeni atâta timp fizic să le facă pe toate! și mai ales să se ia după toate părerile. Cu toții am vrea să ni se explice "pe înțelesul nostru", clar, explicit, fel și fel de lucruri complicate, să învățăm limbi străine în stare de hipnoză, să participăm la politica externă a Statelor Unite. Evident că în această sferă... fiecare este "pe cont propriu" și că există o știință nu numai a prelucrării și difuzării informațiilor (cele adevărate), ci și a protejării lor. Orice firmă își protejează cât mai bine secretele spre exterior, și chiar între angajați, care totuși formează o echipă, mai învață unii de la alții, sau cheamă profesori competenți pentru a le organiza cursuri de formare. Ca profesor de sine stătător, în România, ești obligat să faci aceste lucruri singur, evident, cu un randament mult mai mic.*

Nu am auzit de nici un curs organizat de MEC în care să se predea Java. În schimb, se organizează cursuri de trei lulele pentru promovarea AEL (iată o aplicație "profesionistă" facută de unii care habar nu au ce înseamnă a ține o oră!; totuși, pe acei programatori de trei lulele nu îi "analizează" nimeni, de vină sunt tot profesorii care nu știu să le utilizeze aplicația!!!). Este, de altfel, un fapt binecunoscut că "succesele sunt ale elevilor și eșecurile ale profesorilor".

Și răspunsul, nu se lasă așteptat...

Domnule profesor, nu pot să scriu acest răspuns fără a lua în considerare relațiile noastre de colaborare din trecut și, sperăm noi, și din viitor. Din acest motiv, s-ar putea ca răspunsul meu să nu fie obiectiv. În principiu, aveți dreptate. Nimeni nu are dreptul să ceară doar profesorilor să facă eforturi.

Critica mea nu s-a dorit a fi una feroce ci una constructivă. Am prezentat o alternativă și, mult mai important în opinia mea, am atras atenția asupra unor probleme cauzate de mentalitate (nu doar ale profesorilor).

Nimeni nu vă cere să faceți mai mult decât ceilalți. Eu aș dori ca toată lumea să facă mai mult. Nu vă pot eu spune că e mai bine să învățați de zece ori mai mult decât colegul de catedră care face concursuri de *Minesweeper* în ora de laborator.

Problema este că trebuie să se înceapă de undeva. Eu, datorită specificului acestei reviste, mă pot adresa doar profesorilor de informatică. Nu pot să le cer nici profesorilor de fizică să genereze energie nucleară așa cum nu cer profesorilor de informatică să predea în licee arhitectura procesoarelor *Opteron*. Dar, dacă aș scrie un articol similar la o revistă de fizică, le-aș putea cere profesorilor să renunțe la demonstrațiile matematice pe care oricum nu le înțeleg prea mulți. Există multe de făcut în multe domenii...

Voi încheia cu o scurtă remarcă referitoare la *AEL*. Aplicația este, în opinia mea, sub orice critică deși există unele licee în care este utilizată cu succes. Totuși, vina nu aparține numai programatorilor deoarece dacă ar fi fost o colaborare între profesori și firma producătoare ar fi ieșit cu totul altceva. Dar așa, a ieșit un veritabil produs "*Made in România*".

Vă mulțumesc pentru scrisoare și așteptăm o eventuală "replică la replică".

Vasi Vilvoiu

elev, cls. a VIII-a

Nu sunt un profesor și nici nu am programat în Java. Mă numesc Vasi Vilvoiu și sunt un elev în clasa a 8-a. Alternativa dumneavoastră este foarte interesantă, însă nu cred că se va adopta curând din cauza a două cuvinte: "orientat obiect". Da, este o opțiune demnă de luat în seamă, însă există și VC++ și Delphi (care, apropo, mi se pare un mediu foarte stabil și rapid, care nu e bun numai la "desenarea interfețelor", poate mai bun ca MS VC++) care nu au fost adoptate din același motiv. Lucrul cu obiecte este, într-adevăr, mult mai ușor însă îndepărtează, să zicem, elevul de clasa a 5-a - a 6-a de adevarata structură a algoritmului, chiar din cauza complexității "arborelui genealogic" al claselor și nu numai. Repet, nu spun că este un mediu prost dar, ca multe altele, nu va fi luat în seamă. Aș dori să ținem discuția deschisă. Vă mulțumesc.

Da, dar...

În primul rând trebuie să spun că îmi face o deosebită plăcere să aflu că *GInfo* este citită și de elevii din ciclul gimnazial.

Referitor la opinia ta, trebuie să punctez anumite aspecte. În primul rând, eu nu am propus *Java* din cauză că e un limbaj orientat-obiect. Nu acesta este cel mai important lucru. Algoritmii pot fi predați la fel de bine în *Java* ca și în *C* sau *Pascal*. Este suficient să spui elevului că, în loc de `Program Cmmdc` trebuie să scrie `public class Cmmdc`. În rest mai este funcția `main` în interiorul căreia poate fi scris algoritmul complet fără ca elevul să aibă habar că pe undeva pe acolo sunt și câteva obiecte.

Față de *Microsoft Visual C++* și *Delphi*, *Java* are un avantaj de necontestat: este gratuit și poate rula pe (aproape) orice sistem de operare. Chiar dacă și pentru celelalte medii există versiuni destinate scopurilor educaționale care sunt (aproape) gratuite, atunci când programatorul va folosi mediul respectiv în practică nu va mai beneficia de gratuitate.

Bineînțeles, poate vei spune că pentru majoritatea un astfel de mediu costă între 12.000 și 15.000 de lei (știm noi de ce!), dar atunci când, în sfârșit, se va intra în normalitate nu va mai fi așa.

Îți doresc mult succes în continuare și aștept și alte scrisori.

Un student

Facultatea de Automatică și Calculatoare București

Sunt student în anul doi la Facultatea de Automatică și Calculatoare din București. Mi s-a părut excelent articolul lui Mihai, deși ar mai fi probleme de atins. Să le iau pe rând:

1. Este de-a dreptul trist ce se întâmplă cu învățământul de informatică românesc . Și este și mai trist că toată România se îmbată cu apă rece că avem un învățământ de elită, la nivel european. Ieri am văzut o emisiune la TVR unde se lăuda școala de informatică românească. Nu contestă nimeni valoare elevilor de la olimpiadele internaționale sau a profesorilor lor pregătitori, dar situația la nivel național este cu totul alta. Primul meu contact cu programarea l-am avut în liceu, unde, spre ghinionul meu am avut parte de niște profesori antici, care suspinau încă după Fortran sau după calculatoarele cu cartele perforate, peste care ei domneau cu ceva ani în urmă. Evident că tot ce am învățat a fost de unul singur. Am învățat Pascal și C, m-am dus la olimpiade... am luat și ceva premii. Asta neajutat de nimeni, căci la orice întrebare profesorii mei o

dădeau din colț în colț. Ce pretenții poți să ai de la un profesor care nu știa ce-i un shortcut în Windows sau nu cunoștea diferența dintre QuarkXpress și Quake?

Acum trei ani au ajuns și în fostul meu liceu calculatoarele de la guvern, care cred că și acum sunt ținute la păstrare sub lacăt ca nu cumva să se strice. Deziluzionat de experiența din liceu (apropo, am fost elev la prima clasă din județul Vâlcea, deci nu la un sătuc din vârful muntelui) am decis să-mi încerc norocul la mult galonata Facultate de Calculatoare de la UPB. Aici se pune problema cine rezistă mai mult până să se dea cu capul de pereți. Pe lângă faptul că în ultimii doi ani am făcut mai mult fizică, electronică sau matematici avansate fără nici un fel de folos pentru un viitor informatician, la cele puține materii care aveau legătură cu calculatorul am făcut aceleași chestii învechite de acum 10 ani și cu aceiași profesori bătrîni, incompetenți și corupți. Evident, și aici m-am apucat să învăț singur, deși programul facultății nu-ți lasă prea mult timp liber.

Atunci unde este mărețul nostru învățământ de elită? E mort, îngropat și acum se montează crucea. Și asta este și vina olimpicilor sau a profesorilor lor (din a căror categorie faci și tu parte) care nu fac decât să ridice în slăvi faimosul nostru învățământ, crezând că acesta nu se rezumă decât la mica lor castă. Ce n-aș da să văd un fost olimpic (cum era unul la emisunea de ieri, și care mai este și student la una din faculațile noastre) care să spună la o oră de maximă audiență: "Dotarea este slabă, aproape inexistentă, programa este veche de pe vremea lui Burebista, profesorii sunt bunicii lu' Decebal, sunt depășiți, incompetenți și corupți, iar noi celebri programatori ai României învățăm totul singuri și nu pe la mărețele clădiri comuniste pe care noi le numim facultăți sau colegii naționale." Dar asta nu se va întâmpla niciodată, nu în țara asta.

2. Cât despre Java... Este un limbaj puternic, dar care are și multe părți slabe (cea mai importantă fiind că rulează pe o mașină virtuală, deci nu e foarte rapid). Ar fi foarte util să devină parte a programei de învățământ la clasele mici, pentru ca aceștia să învețe OOP. Între a scrie în Java și a gândi în Java este mare diferență. Cred totuși că la fel de util ar fi să se predea un mediu vizual încă din clasa a IX-a (C++ Builder, Visual C++, Delphi etc.) sau tehnologii pentru internet (HTML, JavaScript, PHP etc.). Aici însă ar trebui făcute multe modificări în programul preuniversitar. Ar fi normal ca un elev de la o clasă de informatică să poată să sară peste orele de biologie sau latină, de exemplu, și să facă informatică. Totul se reduce la "Ar trebui, ar trebui..." însă imediat ne dăm seama că suntem în România, unde nu are cine să predea tehnologii noi și nu există nici calculatoarele pe care să se învețe. Așa că, din păcate, pledoaria ta pentru Java o să rămână doar o simplă pledoarie. Puteai la fel de bine să faci pledoarie și pentru WAP, Python sau C Franklin... tot limbi străine sunt pentru dragii noștri profesori. Dacă nu te apuci chiar tu să predai în școli, atunci o să rămână doar cinci pagini in Gazetă.

3. Deși, pe de o parte faci pledoaria programării în medii vizuale și a programării pe obiecte, pe de cealaltă parte Gazeta te contrazice. 80% din revistă prezintă tot felul de algoritmi care, probabil nu vor fi folosiți de nimeni vreodată. Ar fi mai intersant ca în revistă să scrie despre programare utilă, iar concursul Agora să se transforme într-un concurs de soft la nivel național. În condițiile în care revista prezintă doar chestii teoretico-filozofice și prea puține lucruri utile este normal că are prețul pe care îl are și este nevoie de sponsori pentru tipărirea ei. Mai cinstit ar fi să o mutați pe Web dacă oricum nu scoate profit. Totuși ar fi bine dacă aceasta s-ar transforma într-o revistă de programare utilă. În România nu există nici o astfel de publicație. În afară sunt zeci (reviste de C, Php, Java etc.). Lipsa unei astfel de reviste este încă o pată negră pe și așa pătatul obraz al învățământului românesc de informatică.

Cam atât pentru moment, deși aș putea să vorbesc despre subiectul acesta o zi. Oricum, felicitări pentru articol și sper să mai fie oameni ca tine care să spună nu numai ce vor cei mari să audă.

Nici chiar așa...

Salut! Trebuie să recunosc că scrisoarea ta a stârnit câteva zâmbete în redacție. Din nefericire, a cam fost haz de necaz... Totuși, părarea mea (și a întregii redacții) este că exagerezi. Dacă ai auzit despre un profesor corupt, nu înseamnă că toți sunt corupți. Chiar dacă ai avut un profesor nepriceput, ar trebui să-i numeri și pe cei pricepuți. Nu e totul chiar atât de negru cum ai dat tu de înțeles. Nu toți olimpicii învață singuri! Există doi-trei-zece profesori în țara asta care chiar se pricep la ceea ce fac. Problema e că ar trebui să fie cel mult doi-trei-zece care nu se pricep!

Afirmi că pledoaria mea va rămâne doar o pledoarie și ai dreptate. Nu pot să îmi fac iluzii că un astfel de articol scris într-o revistă care oricum nu este citită de cei care ar putea face schimbări, va genera o "revoluție". Totuși, consider că este de datoria mea să atrag atenția.

Dacă totul va rămâne în două-trei numere din *GInfo*, voi avea satisfacția că am încercat și poate voi dormi mai liniștit. Dacă anumiți cititori vor ridica problema în alte "sfere", atunci poate că nu voi dormi la fel de liniștit. Oricum, nu îmi stă în fire să nu spun ce gândesc! Și eu consider că *Pascal*-ul e depășit și limbajul *Java* este mai ușor de învățat decât limbajul *C*. Este părerea mea și vreau să cred că și ea contează. Până acum nu am primit nici o replică despre care aș putea spune că m-a pus pe gânduri. Cred că este momentul ca învățământul românesc de informatică să meargă mai departe.

La final...

Ați putut citi părerile celor două "tabere" (doi profesori și doi "beneficiari" ai sistemului de învățământ). Se observă destul de clar că profesorii sunt reticenți la propunerea de utilizare a limbajului *Java*, în timp ce ceilalți cochetează cu această idee.

Concluzia cred că o puteți trage singuri. Oricum, noi așteptăm în continuare scrisorile voastre. Discuția va continua atâta timp cât cineva va dori să își exprime părerea.

12-14 decembrie 2003, Râmnicu Vâlcea

Concursul de Informatică "NICOLAE PĂUN"

În perioada 12-14 decembrie 2003 a avut loc la Râmnicu Vâlcea prima ediție a Concursului Interjudețean de Informatică "Nicolae Păun". În continuare vă prezentăm enunțurile celor șase probleme propuse spre rezolvare.

Clasa a IX-a

P010401: Grădina

Nepotul unui grădinar vine să-și petreacă vacanța. Într-o zi acesta merge cu grădinarul la muncă. Grădina pe care trebuie să o îngrijească are o suprafață pătratică. Toate plantele mai speciale au fost plantate în puncte de coordonate întregi (sistemul de coordonate are originea în centrul pătratului). Nepotul, copil neastâmpărat, începe să se joace și la un moment dat pornește din greșeală mașina de tuns iarba, care începe să taie toate plantele pe traiectorii circulare concentrice cu centrul în centrul pătratului.

Se știe că în fiecare punct de coordonate întregi se află plante speciale. Razele traiectoriilor mașinii de tuns iarba se construiesc cu ajutorul a două numere naturale R și X, cu semnificația că traiectoriile circulare au razele: R, $R + X$, $R + 2 \cdot X$, $R + 3 \cdot X$ etc. Mașina de tuns iarba se oprește când trece pe o traiectorie circulară care conține în interior toată grădina.

Se cere numărul de plante speciale distruse de către mașina de tuns iarba.

Date de intrare

Din fișierul standard de intrare se citesc de pe un singur rând, separate prin câte un spațiu, trei numere naturale L, R și X care reprezintă latura pătratului, raza primei traiectorii circulare a mașinii de tuns iarba și valoarea care va fi utilizată la construirea razelor celorlalte traiectorii circulare, așa cum este descris în enunț.

Date de ieșire

La ieșirea standard se va afișa un singur număr natural care reprezintă numărul de plante speciale distruse de către mașina de tuns iarba.

Restricții și precizări

- L, R și X sunt numere naturale nenule mai mici decât 1000;
- se consideră că traiectoria mașinii de tuns iarba nu are grosime;
- trecerea de pe o traiectorie circulară pe următoarea se face prin zone care nu conțin plante speciale;
- frontiera grădinii face parte din grădină.

Exemplu

Intrarea standard	*Ieșirea standard*
4 1 1	8

Timp maxim de execuție/test: 1 secundă

P010402 Numărare

Se consideră un număr natural N din intervalul [1, 1000].

Să se determine câte numere naturale de N cifre cu produsul cifrelor egal cu 8 există.

Date de intrare

Din fișierul standard de intrare se citește de pe un singur rând, numărul natural N.

Date de ieșire

La ieșirea standard se va afișa un singur număr natural care reprezintă numărul de numere naturale cu N cifre pentru care produsul cifrelor lor este egal cu 8.

Exemplu

Intrarea standard	*Ieșirea standard*
2	4

Timp maxim de execuție/test: 3 secunde

Clasa a X-a

P010403: Ferme

La marginea unui sat se află două ferme înconjurate de garduri în formă de poligoane convexe. În jurul fermelor se găsește o mare pășune. Proprietarii celor două ferme vor să aibă o mică potecă, pentru a face legătura între ferme, dar să distrugă cât mai puțin din pășune. Fiecare extremitate a potecii poate fi orice punct situat la marginea oricărei ferme. Lungimea potecii este distanța dintre extremitățile sale.

Determinați cea mai scurtă potecă de la prima la cea de-a doua fermă.

Date de intrare

Fișierului de intrare **FERME.IN** conține pe prima linie un număr natural nenul, $K1$ care reprezintă numărul de colțuri ale gardului primei ferme.

Cea de-a doua linie conține $K1$ perechi de numere întregi, $x_1\ y_1\ x_2\ y_2\ \ldots\ x_{k1}\ y_{k1}$, separate prin câte un spațiu, reprezentând coordonatele vârfurilor gardului primei ferme.

Pe cea de-a treia linie se află numărul natural nenul, $K2$, reprezentând numărul de colțuri ale gardului celei de-a doua ferme.

Cea de-a patra linie conține $K2$ perechi de numere întregi, $x_1\ y_1\ x_2\ y_2\ \ldots\ x_{k2}\ y_{k2}$, separate prin câte un spațiu reprezentând coordonatele vârfurilor gardului celei de-a doua ferme.

Date de ieșire

În fișierul de ieșire **FERME.OUT** se vor scrie, pe o singură linie, patru numere reale cu patru zecimale, separate prin câte un spațiu, reprezentând extremitățile potecii de legătură determinate.

Restricții și precizări

- $2 \le K1, K2 \le 20$;
- coordonatele vârfurilor gardurilor care înconjoară prima fermă, respectiv pe cea de-a doua, sunt numere întregi cuprinse între -100 și 100 și sunt date fie în ordinea deplasării acelor de ceasornic, fie în sens trigonometric;
- cele două ferme nu au nici un punct comun (nu au puncte interioare comune și nu au puncte comune pe gardurile lor).

Exemplu

FERME.IN

```
4
3 4 3 2 5 2 4
4
8 3 8 6 11 6 11 3
```

FERME.OUT

```
5.0000 3.5000 8.0000 3.5000
```

Timp maxim de execuție/test: 0.5 secunde

P010404: Young

Se consideră m numere naturale $n_1, n_2, \ldots, n_m$, cu proprietatea că: $n_1 \ge n_2 \ge \ldots \ge n_m > 0$. Se numește *tablou Young* o aranjare a $n_1 + n_2 + \ldots + n_m$ numere naturale nenule distincte într-un tablou, astfel încât pe fiecare linie i sunt n_i elemente ($1 \le i \le m$) în ordine crescătoare de la stânga la dreapta, iar elementele de pe aceeași coloană sunt în ordine crescătoare de sus în jos.

Un exemplu de *tablou Young* pentru $m = 4$, $n_1 = 6$, $n_2 = 4$, $n_3 = 4$, $n_4 = 1$ este următorul:

```
 1   2   5   9  10  15
 3   6   7  13
 4   8  12  14
11
```

Dându-se $n_1, n_2, \ldots, n_m$, să se determine câte *tablouri Young* formate din elementele $1, 2, \ldots, n_1 + n_2 + \ldots + n_m$ se pot forma.

Date de intrare

Pe prima linie a fișierului de intrare **YOUNG.IN** se află numărul natural m. Pe cea de-a doua linie a fișierului se află numerele $n_1, n_2, \ldots, n_m$ separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire **YOUNG.OUT** va conține numărul de *tablouri Young* care se pot forma.

Restricții

- $1 \le m \le 6$;
- $n_i \le 6$, $1 \le i \le m$.

Exemplu

YOUNG.IN

```
2
3 2
```

YOUNG.OUT

```
5
```

Explicație

Cele 5 tablouri Young sunt:

```
1 3 5    1 2 3    1 2 4    1 3 4    1 2 5
2 4      4 5      3 5      2 5      3 4
```

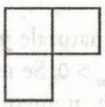
Timp maxim de execuție/test: 1 secundă

Clasele a XI-a și a XII-a

P010405 Parcele

O suprafață agricolă pătratică este împărțită prin drepte orizontale și verticale în $(6 \cdot N + 1)^2$ parcele pătratice egale. În figura 1 se poate observa suprafața obținută pentru $N = 1$.

Suprafața agricolă este dată în arendă țăranilor din sat și este parcelată (fiecare țăran primește o astfel de parcelă) din nou în parcele de forma:



Parcelarea se realizează astfel încât:
- să nu existe suprapuneri ale acestor parcele;
- să rămână nedată țăranilor suprafața din parcela pătratică de pe linia P și coloana Q (unde va fi o fântână publică).

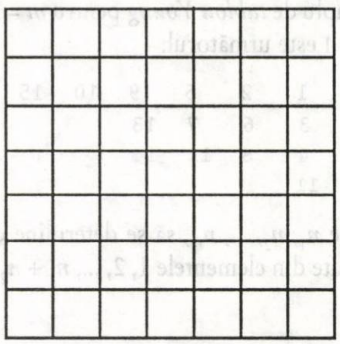


Figura 1

Date de intrare
Prima linie a fișierului de intrare **PATRAT.IN** conține numărul natural nenul N, având semnificația din enunț, iar următoarea linie conține două numere naturale nenule, P și Q, având semnificația din enunț.

Date de ieșire
În fișierul de ieșire **PATRAT.OUT** se vor afișa 1, 2, ..., $6 \cdot N + 1$ linii. Pe fiecare din aceste linii se vor afla $6 \cdot N + 1$ valori întregi pozitive. Ele corespund unui tablou bidimensional cu $6 \cdot N + 1$ linii și $6 \cdot N + 1$ coloane care descrie modul de parcelare.

Pentru fiecare parcelă există trei numere egale (din mulțimea $\{1, 2, ..., 12 \cdot N^2 + 4 \cdot N\}$) ce o reprezintă.

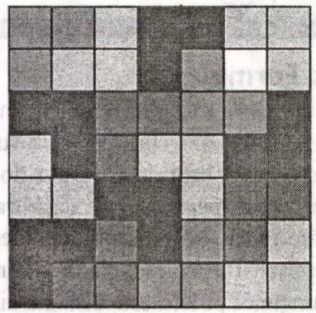
Nu se pot codifica două parcele cu același număr.

Parcelei pătratice de pe linia P și coloana Q îi va fi asociat numărul 0.

Restricții și precizări
- $1 \le N \le 20$;
- $1 \le P, Q \le 6 \cdot N + 1$;
- ordinea de numerotare a parcelelor nu este importantă.

Exemplu
PATRAT.IN
```
1
2 6
```

PATRAT.OUT
```
1 1 2 3 3 5 5
1 2 2 3 4 0 5
6 6 8 8 4 4 10
7 6 8 9 9 10 10
7 7 12 12 9 16 16
13 13 14 12 15 16 11
13 14 14 15 15 11 11
```



Timp maxim de execuție/test: 0,5 secunde

P010406: Ghicit
Se consideră un număr natural n. Se cere să se determine n vectori $v_1, v_2, ..., v_n$ fiecare având 2^{n-1} componente distincte cu proprietatea că orice număr natural nenul mai mic sau egal cu 2^n-1 poate fi *ghicit* știind în care din vectori se află. Mai precis, dacă un număr se află în vectorii $v_{i_1}, v_{i_2}, ..., v_{i_k}$, atunci acesta este egal cu suma dintre cele mai mici elemente conținute de acești vectori.

Date de intrare
Fișierul text **GHICIT.IN** va conține o singură linie pe care se va afla numărul natural n.

Date de ieșire
Fișierul text **GHICIT.OUT** va conține pe prima linie prima componentă din fiecare din vectorii $v_1, v_2, ..., v_n$ separate prin câte un spațiu. Pe a doua linie, cea de-a doua componentă din fiecare din vectorii $v_1, v_2, ..., v_n$ separate prin câte un spațiu ș.a.m.d.

Restricție
- $1 \le n \le 20$.

Exemplu
GHICIT.IN
```
3
```

GHICIT.OUT
```
1 5 6
3 7 3
7 6 2
5 4 7
```

Explicație
Soluția este corectă pentru că oricare număr natural dintre 1, 2, 3, 4, 5, 6, 7, poate fi *ghicit* dacă se știe în care din cei trei vectori se află. Spre exemplu, pentru numărul 5, acesta poate fi *ghicit* știind că se află în primul și în al doilea vector din fișierul **GHICIT.OUT**, pentru că 5 = 1 + 4 (suma dintre cel mai mic element din primul vector, respectiv al doilea vector).

Timp maxim de execuție/test: 1 secundă

Craiova, 2003

Concursul de Informatică "ȘTEFAN ODOBLEJA"

În continuare vă vom prezenta enunțurile celor nouă probleme propuse spre rezolvare la Concursul de informatică "Ștefan Odobleja" organizat în primăvara anului trecut la Craiova. Problemele prezentate mai jos sunt atât cele pentru proba individuală, cât și cele pentru proba pe echipe. Redacția Ginfo a modificat ușor enunțurile pentru a oferi cititorilor enunțuri mai concise.

Clasa a IX-a

P010407: Dilema furnicuțelor

În curtea din spatele casei unui gospodar trăiau nestingherite mai multe familii de furnicuțe, organizate în mai multe mușuroaie.

Una dintre furnicuțe îl aude într-o zi pe gospodar spunându-i soției sale că vrea să amplaseze un obiect chiar în zona în care se aflau mușuroaiele.

Furnicuța le anunță și pe celelalte, însă neștiind să aprecieze exact unde va fi amplasat acel obiect, furnicuțele au nevoie de ajutorul vostru. Dacă află care din mușuroaie se află în pericol, ele se pot adăposti fără grijă în cele care sunt în siguranță.

Cunoscându-se numărul de puncte (mușuroaie) M, coordonatele x și y ale celor M puncte prin numărul vârfurilor poligonului (obiectului) și coordonatele vârfurilor, determinați mușuroaiele care sunt în pericol să fie distruse de obiectul care va fi amplasat de gospodar.

Date de intrare

Fișierul de intrare **furnici.in** conține pe prima linie un număr întreg pozitiv M, care reprezintă numărul de mușuroaie.

Pe următoarele M linii se găsesc câte două numere întregi pozitive, separate printr-un singur spațiu, care reprezintă coordonatele punctelor (mușuroaielor).

Următoarea linie conține un singur număr întreg pozitiv N care reprezintă numărul de vârfuri ale poligonului.

Cele N linii care urmează conțin câte două numere întregi pozitive, separate prin câte un spațiu, care reprezintă coordonatele pentru vârfurile poligonului. Poligonul poate fi convex sau concav.

Date de ieșire

Fișierul de ieșire **furnici.out** va conține M linii. Pe fiecare dintre acestea se va scrie unul dintre mesajele "`in siguranta!`" sau "`in pericol`" dacă mușuroaiele de furnici din fișierul de intrare se găsesc în exteriorul, respectiv în interiorul poligonului.

Punctele corespunzătoare mușuroaielor vor fi tratate în ordinea în care apar în fișierul de intrare.

Restricții și precizări

- se consideră că un mușuroi este în pericol atunci când coordonatele sale sunt conținute în interiorul poligonului sau când mușuroiul se găsește pe una dintre laturile acestuia;
- $3 \le N \le 50$;
- $1 \le M \le 100$.

Exemplu

furnici.in

```
5
80 100
400 170
230 160
340 347
230 350
6
100 165
380 200
400 140
520 350
395 310
220 360
```

furnici.out

```
in siguranta!
in pericol!
in siguranta!
in siguranta!
in pericol
```

P010408: Verde la semafor

Lucrând în tura de noapte, *Ovidiu* pleacă din parcarea fabricii la care este angajat la ora 02:00 *AM* fix, în fiecare dimineață.

Drumul său spre casă este drept și de-a lungul lui sunt plasate unul sau mai multe semafoare. *Ovidiu* s-a întrebat adesea dacă, dându-se locațiile exacte ale semafoarelor și perioadele de timp la care se schimbă culoarea acestora, există viteze cu care să meargă acasă fără să fie nevoit să accelereze sau să încetinească în funcție de culoarea roșie a semaforului.

Pentru a-și satisface curiozitatea, el vă roagă să faceți un program care să calculeze aceste viteze.

Programul ar trebui să găsească toate vitezele (numere întregi, pozitive), în kilometri pe oră, pe care să le urmeze *Ovidiu* în drumul său spre casă.

Fiecare viteză (în kilometri pe oră) trebuie menținută de el din momentul în care pleacă din parcare la ora 02:00 *AM* până ajunge acasă, astfel încât el să nu treacă niciodată pe culoarea roșie a semaforului.

Ovidiu poate trece de un semafor chiar în momentul în care acesta își schimbă culoarea din galben în roșu, sau chiar în momentul în care roșu se transformă în verde.

Datorită faptului că *Ovidiu* este un cetățean care nu vrea să încalce prea mult legea, trebuie considerate numai vitezele mai mici sau egale cu 60 km/h. El nu vrea însă nici să meargă foarte încet, de aceea nu trebuie considerate nici vitezele mai mici de 30 km/h.

Ovidiu a aflat dintr-o sursă sigură faptul că toate semafoarele se schimbă în verde la ora 02:00 *AM*.

Date de intrare

Fișierul de intrare **semafoare.in** conține una sau mai multe mulțimi de date care descriu semnale de trafic. Lista de mulțimi este urmată de valoarea `-1` pe o singură linie.

O mulțime este dată pe mai multe linii. Astfel, prima linie a unei mulțimi conține un singur număr natural N care reprezintă numărul de semafoare de pe traseu.

Pe următoarele N linii corespunzătoare unei mulțimi se află patru numere reale pozitive L, V, G și R separate între ele prin spații. L reprezintă distanța la care este plasat un semafor, în kilometri, față de parcare. V, G și R sunt perioadele de timp exprimate în secunde până la schimbarea culorilor *verde*, *galben*, respectiv *roșu* ale semafoarelor.

Date de ieșire

Fișierul de ieșire **semafoare.out** constă din tratarea fiecărei mulțimi în parte, urmată de o listă a tuturor vitezelor care sunt potrivite pentru ca *Ovidiu* să evite culoarea roșie.

Vitezele, care sunt consecutive, ar trebui să fie specificate ca intervale de forma m-M, unde m și M sunt vitezele cele mai mici, respectiv cele mai mari din interval. Intervalele de forma M-M (adică de lungime 1) trebuie scrise doar ca M. Intervalele trebuie să fie separate prin virgule.

Dacă nu există viteze care să îndeplinească cerințele, în fișierul de ieșire se va scrie `"Nu exista nici o viteza"`.

Pentru fiecare mulțime din fișierul de intrare, în fișierul de ieșire se va scrie o singură linie. La începutul unei linii se va scrie mesajul `"Cazul X: "`, unde X este numărul de ordine al mulțimii din fișierul de intrare. Acest mesaj este urmat de un spațiu și fie de mesajul `"Nu exista nici o viteza"`, fie de o listă de viteze separate prin caracterul `", "` așa cum indică exemplul prezentat în continuare.

Restricții și precizări

- numărul N de semafoare are cel mult valoarea 6;
- distanța unui semafor față de fabrică este un număr real și nu va depăși valoarea 32000;
- intervalul de timp care se scurge de la aprinderea unei culori a unui semafor și până la schimbarea acesteia este dat de un număr întreg a cărui valoare este cuprinsă între 1 și 256;
- numărul de mulțimi din fișierul de intrare nu va depăși valoarea 100.

Exemplu

semafoare.in

```
1
5.5 40 8 25
3
10.7 10 2 75
12.5 12 5 57
17.9315 4 67
-1
```

semafoare.out

```
Cazul 1: 30, 32-33, 36-38, 41-45, 48-54, 59-60
Cazul 2: Nu exista nici o viteza.
```

Clasa a X-a

P010409: Traiectoria bilei

Se consideră o masă dreptunghiulară de dimensiune $M \times N$ (asemănătoare cu cea de biliard), cu o gaură în interior.

Pe masă se află o bilă (se va considera de dimensiune neglijabilă), care se deplasează în interiorul acestei mese și care se poate lovi de marginile acesteia sau poate cădea în gaura care se află pe masă după cum se poate observa în figura 1.

Ciocnirile cu manta (marginile mesei) se fac cu un unghi de incidență egal cu unghiul de reflexie.

Traiectoria pornește dintr-un punct inițial dat (x, y). Se precizează și direcția în care aceasta pornește. Traiectoria se calculează până când s-a efectuat un număr de ciocniri dat sau până când bila cade în gaură.

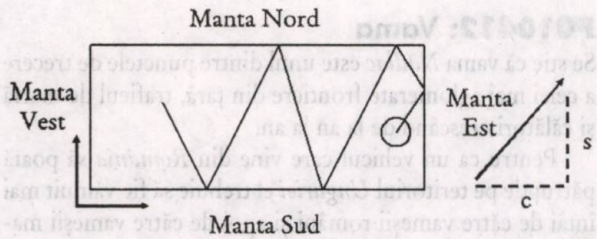


Figura 1

Să se calculeze traiectoria bilei care se deplasează în interiorul acestei mese.

Date de intrare

Fișierul de intrare **BILA.IN** va conține pe prima linie numărul de ciocniri pentru care se calculează traiectoria.

Pe cea de-a doua linie se află două numere reale M și N, separate printr-un singur spațiu, care reprezintă dimensiunile mesei.

Ce-a de-a treia linie conține trei numere separate între ele prin spații care reprezintă coordonatele centrului găurii circulare și raza acesteia.

Cea de-a patra linie conține patru numere separate între ele printr-un singur spațiu, care indică poziția și direcția inițiale ale bilei.

Direcția inițială a bilei este dată de două numere (c, s), care sunt lungimile proiecțiilor unei săgeți de lungime 1 (de-a lungul direcției de mișcare a bilei) pe axele Ox și Oy. Cu alte cuvinte, c și s sunt *cosinusul* și *sinusul* unghiului măsurat în sens invers acelor de ceasornic de pe direcția $+Ox$. Dacă aceste proiecții sunt în direcțiile $-Ox$ sau $-Oy$, numerele respective sunt negative.

Trebuie tratat și cazul în care bila ajunge într-un colț, situație în care se întoarce pe aceeași linie pe care a venit.

Date de ieșire

În fișierul de ieșire **BILA.OUT** se va scrie câte o linie pentru fiecare ciocnire de pe traiectorie, identificând numărul ciocnirii, manta (*Nord*, *Sud*, *Est*, *Vest*) cu care s-a ciocnit bila și coordonatele de-a lungul mantei ale punctului de ciocnire așa cum apare în exemplul prezentat în continuare.

Dacă ciocnirea are loc într-un colț, se va afișa pe o linie, despre care colț este vorba (*SudEst*, *SudVest*, *NordEst*, *NordVest*). Spre exemplu, dacă la a treia ciocnire cu bila a ajuns în colțul *NordEst* se va afișa: `"ciocnirea 3 in coltul NordEst"`.

Liniile traiectoriei se vor genera până când se atinge numărul maxim de ciocniri sau până în momentul în care bila cade în gaură.

Bila cade în gaură dacă traiectoria calculată intersectează cercul care reprezintă suprafața găurii. În acest caz în fișierul de ieșire se va scrie pe o singură linie mesajul `"Bila a cazut in gaura!"`.

Restricții și precizări

- numărul de ciocniri cu manta este un număr întreg pozitiv cu valoarea cuprinsă între 1 și 100;
- raza găurii este un număr întreg pozitiv a cărui valoare este cuprinsă între 1 și 100;
- $4 \le N, M \le 1.000$.

Exemplu

BILA.IN

```
12
100 60
70 40 10
14 35 0.6 0.8
```

BILA.OUT

```
ciocnirea 1 la zidul Nord x = 32.7500
ciocnirea 2 la zidul Sud x = 77.7500
ciocnirea 3 la zidul Est y = 29.6667
ciocnirea 4 la zidul Nord x = 77.2500
Bila a cazut in gaura!
```

Timp de execuție: 1 secundă/test

P010410: Comoara

Un grup de arheologi format din K persoane au găsit o comoară compusă din M lingouri de aur pe care doresc să le împartă fără să le secționeze.

Există mai multe posibilități de împărțire a acestora.

Există posibilitatea ca la împărțire unele persoane să nu primească nici un lingou.

La rugămintea arheologilor, sarcina voastră este să găsiți numărul total de modalități de a împărți comoara, astfel încât, la fiecare împărțire, fiecare persoană să primească un număr diferit de lingouri.

Date de intrare

Fișierul de intrare **comoara.in** conține pe o singură linie două numere întregi M și K, separate printr-un singur spațiu, reprezentând numărul de lingouri, respectiv numărul de persoane.

Date de ieșire

Fișierul text **comoara.out** conține pe o singură linie numărul de modalități de a împărți comoara astfel încât fiecare persoană să primească un număr diferit de lingouri la fiecare împărțire.

Restricții

- $1 \le M \le 1.000$;
- $1 \le K \le 50$.

Exemplu

comoara.in

```
20 4
```

comoara.out

```
11
```

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P010411: Locuințe

În ținutul *Cucuberland* populația este în continuă creștere. Suprafața ținutului este destul de mică, foarte curând putându-se manifesta o criză a locuințelor. Pentru aceasta, primăria a luat decizia să construiască locuințe într-o regiune mlăștinoasă, întinsă ca suprafață, dar inutilizabilă din cauza prezenței apei. Mlaștina are forma unui pătrat cu latura de lungime N metri.

Deoarece costul asanării mlaștinii era prea mare, arhitecții au propus amplasarea unor piloni, pe aceștia urmând a se construi locuințe, fără a mai fi necesară asanarea mlaștinii. Pilonii vor fi amplasați într-o rețea rectangulară la distanța de 1 m pe verticală sau orizontală. În figura următoare este prezentată o rețea de piloni pentru o mlaștină cu latura de 2 metri.

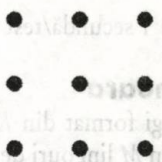


Figura 2

Se știe că o locuință are, de asemenea o formă pătratică, iar pentru a putea fi amplasată, toate cele patru colțuri ale sale trebuie să se sprijine pe câte un pilon.

Pentru a putea aproba planul de urbanism, primarul trebuie să cunoască numărul total de modalități în care poate fi amplasată o locuință pătrată în zona mlăștinoasă respectivă.

Date de intrare

Pe prima linie a fișierului de intrare **locuinte.in** se află un singur număr natural N reprezentând latura suprafeței mlăștinoase.

Date de ieșire

Fișierul de ieșire **locuinte.out** va conține un număr natural corespunzător numărului de modalități în care poate fi amplasată o locuință de formă pătrată.

Restricții și precizări

- $1 \le N \le 1.000.000.000$;
- lungimea laturii locuinței este un număr întreg cu valoarea cuprinsă între 1 și N.

Exemplu

locuinte.in

```
2
```

locuinte.out

```
6
```

Timp de execuție: 1 secundă/test

P010412: Vama

Se știe că vama *Nădlac* este unul dintre punctele de trecere a celei mai aglomerate frontiere din țară, traficul de marfă și călători crescând de la an la an.

Pentru ca un vehicul care vine din *România* să poată pătrunde pe teritoriul *Ungariei* el trebuie să fie vămuit mai întâi de către vameșii români și apoi de către vameșii maghiari, în această ordine. În vama română, într-un schimb se află M lucrători, iar în vama ungară P lucrători.

Un vehicul, odată ajuns în vamă, așteaptă să fie preluat de către un vameș român. După ce trece de vama română, acesta va trebui să parcurgă aceleași operații în vama ungară.

Un vameș controlează la un moment dat cel mult un vehicul. Nu toți vameșii lucrează în același ritm.

Un vameș controlează o mașină indiferent de tipul sau încărcătura acesteia în aceeași perioadă de timp cunoscută.

La miezul nopții în vama română se află un număr N de vehicule care așteaptă să ajungă pe teritoriul *Ungariei*.

Pentru fluidizarea traficului în vamă calculați timpul minim necesar pentru ca toate cele N vehicule să fie vămuite la punctul de trecere al frontierei *Nădlac*.

Date de intrare

Pe prima linie a fișierului de intrare **vama.in** se află, pe o singură linie, trei numere întregi N, M și P, unde N reprezintă numărul de vehicule care se află în așteptare la miezul nopții în vamă, M reprezintă numărul de lucrători din vama română, iar P reprezintă numărul de vameși din vama ungară. Linia a doua conține M numere întregi, separate prin câte un spațiu, care reprezintă timpii de verificare al unui vehicul corespunzător fiecărui vameș român în parte. Pe cea de-a treia linie a fișierului de ieșire sunt scrise P numere întregi, separate prin câte un spațiu, care reprezintă timpii de verificare al unui vehicul corespunzător fiecărui vameș ungur în parte.

Date de ieșire

Fișierul de ieșire **vama.out** va conține un număr întreg corespunzător timpului minim de vămuire a tuturor celor N vehicule.

Restricții și precizări

- $1 \le N \le 1.000$;
- $1 \le M \le 100$;
- $1 \le P \le 100$.

Exemplu

vama.in

```
6 3 2
1 1 10
2 3
```

vama.out

```
9
```

Timp de execuție: 1 secundă/test

P010413: Mobuto

O bună parte din suprafața țării *Mobuto* este acoperită de junglă. Pentru a se mișca eficient prin junglă dintr-o localitate în alta, băștinașii se deplasează în "șir indian".

Din localitatea *Mbengdu* pleacă spre localitatea *Mulato* o delegație de *N* persoane. În dimineața celei de-a noua zile, șeful delegației observă că membri acesteia suferă de "stresul vecinului din față": fiecare s-a săturat să tot vadă aceeași ceafă roșie.

Pentru a preîntâmpina o revoltă, acesta hotărăște să schimbe locurile persoanelor din delegație astfel încât nici una să nu-l mai vadă pe vecinul de până acum.

Scrieți un program care determină numărul de posibilități pe care le are la dispoziție șeful delegației.

Date de intrare

Fișierului de intrare **mobuto.in** conține, pe o singură linie, un numărul natural *N* care reprezintă numărul persoanelor din delegație.

Date de ieșire

Fișierul de ieșire **mobuto.out** va conține, pe o singură linie, un singur număr natural, corespunzător numărului de modalități în care se poate schimba configurația șirului indian conform cerinței.

Restricție

- $1 < N < 100$.

Exemplu

mobuto.in

```
3
```

mobuto.out

```
3
```

Timp de execuție: 1 secundă/test

P010414: Spreadsheet

Un *spreadsheet* este un tablou dreptunghiular de celule după cum se poate observa în figura 3. O celulă conține o valoare sau o formulă care este o expresie.

Un *spreadsheet* "simplu" este unul în care valorile din celule sunt numere întregi și expresiile pot fi sume, diferențe, produse, câturi de întregi și/sau referințe la alte celule.

Dacă o expresie referă un singur întreg, atunci ea este înlocuieeetă de valoarea acesteia.

Să se realizeze un program care evaluează valorile fiecărei celule.

Date de intrare

Fișierul de intrarea **calcule.in** conține un astfel de *spreadsheet*. Un *spreadsheet* poate conține cel mult 256 linii și 256 coloane după cum indică figura 3. Coloanele sunt etichetate cu litere mari *A*, *B*, ..., *Z*, *AA*, *AB*, ..., *IV*. Liniile sunt etichetate cu numere de la 1 la 256. O celulă este identificată prin specificarea coloanei urmată de numărul liniei, aceasta numindu-se referință.

	A	B	C	...	Z	AA	AB	...	AZ	BA	...	IV
1												
2												
3												
...												
256												

Figura 3

Astfel, prima celulă din colțul stânga-sus este *A*1, iar ultima celulă din colțul dreapta-jos este *IV*256. Dacă o celulă conține o constantă, atunci aceasta se specifică prin precizarea valorii acesteia, iar dacăo celulă conține o formulă, atunci aceasta se precizează prin semnul "=" urmat de expresia formulei.

Fișierul de intrare va conține linii de forma:

- `referință_celulă valoare`

  sau
- `referință_celulă =expresie`

Celulele a căror valoare nu se specifică se consideră egale cu 0.

O formulă poate conține întregi, referințe la alte celule și operatorii +, -, *, / reprezentând adunarea, diferența, înmulțirea și împărțirea, precum și paranteze ( ).

Se consideră că toate constantele și formulele sunt corecte din punct de vedere sintactic.

O expresie are următoarea sintaxă:

- `expresie:= referință_celulă`
- `expresie:= constantă`
- `expresie:= (expresie)`
- `expresie:= expresie + expresie`
- `expresie:= expresie - expresie`
- `expresie:= expresie * expresie`
- `expresie:= expresie / expresie`
- `expresie:= -expresie`
- `constantă:= număr_întreg_cu_semn`

Date de ieșire

Fișierul de ieșire **calcule.out** va conține pe câte o linie valorile pentru celulele date în fișierul de intrare în aceeași ordine sub următoarea formă:

`referință_celulă valoare`.

Dacă valoarea unei celule nu se poate evalua, în locul acesteia se va afișa `###`.

Operațiile din cadrul unei expresii se evaluează în ordine de la stânga la dreapta ținându-se cont de prioritate, care este cea obișnuită.

Restricții și precizări

- constantele sunt numere întregi cuprinse între -32768 și 32767;

- nici o expresie nu poate conține mai mult de 75 de caractere;
- nici o expresie nu conține spații în interior.

Exemplu

calcule.in

```
A2 =B1
B1 3
C8 =C1+B2
C1 =C8
AA25 =-B1+(A2/2)
```

calcule.out

```
A2 3
B1 3
C8 ###
C1 ###
AA25 -2
```

Timp de execuție: 1 secundă/test

P010415: Gardian

O galerie de artă are forma unui poligon. Datorită faptului că bugetul alocat întreținerii galeriei de artă este redus, directorul acesteia dorește să angajeze un singur gardian care să asigure paza imobilului.

Supraveghetorul trebuie să patruleze într-un perimetru, astfel încât în orice moment să poată observa fiecare latură a încăperii.

În figura 4, se află un exemplu de poligon, iar zona hașurată reprezintă suprafața în care supraveghetorul poate patrula astfel încât să observe fiecare latură a încăperii.

Se cere să se realizeze un program care să determine această suprafață.

Date de intrare

Pe prima linie a fișierului de intrare **gardian.in** se află numărul N de colțuri ale încăperii.

Următoarele N linii conțin două numere x și y, separate printr-un singur spațiu, reprezentând coordonatele unui colț al încăperii.

Coordonatele unui colț k sunt date de două numere întregi x_k, y_k, separate printr-un spațiu.

Colțurile sunt date în ordine trigonometrică.

Date de ieșire

Fișierul de ieșire **gardian.out** va conține pe prima linie numărul M de colțuri ale suprafeței cerute.

Pe următoarele M linii se vor scrie coordonatele suprafeței sub forma x_k, y_k, unde x_k și y_k sunt două numere reale afișate cu două zecimale exacte și vor fi separate printr-un spațiu.

Colțurile vor fi scrise în ordinea crescătoare a coordonatei x_k, iar pentru valori egale ale acesteia, în ordinea crescătoare a coordonatei y_k.

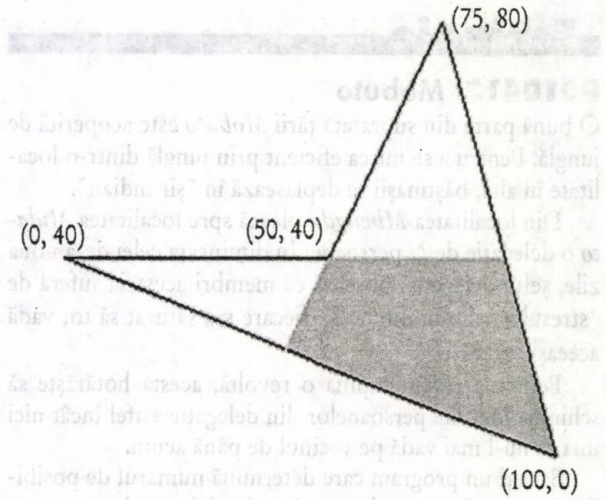


Figura 4

Dacă zona de supraveghere este vidă, atunci fișierul de ieșire va conține doar numărul 0.

Restricții

- $3 \le N < 101$;
- $-32000 < x_k, y_k < 32000$, $1 \le k \le N$.

Exemplu

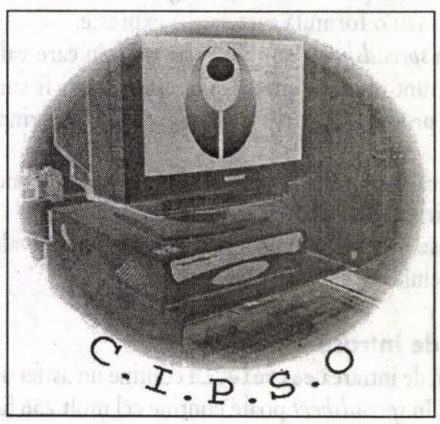
gardian.in

```
4
100 0
75 80
50 40
0 40
```

gardian.out

```
4
40.00 24.00
50.00 40.00
87.50 40.00
```

Timp de execuție: 1 secundă/test



Iași, România

Olimpiada BALCANICĂ de informatică

În continuare vă vom prezenta soluțiile celor șase probleme propuse la cea de-a XI-a ediție a Olimpiadei Balcanice de Informatică. Aceste soluții au fost realizate de redacția Ginfo pe baza soluțiilor oficiale prezentate de către autorii problemelor.

P060313: Secvența Farey

Pentru a rezolva problema, vom aborda următoarea subproblemă:

"Dându-se o fracție A / B, nu neapărat ireductibilă, să se afle câte fracții din secvența Farey sunt mai mici decât ea."

Având o funcție care rezolvă această subproblemă, algoritmul pentru problema din concurs ar fi următorul:

- determinăm numărul X astfel încât răspunsul să fie între X/N și $(X + 1)/N$, folosind căutarea binară pentru X și apelând funcția de mai sus de $\log N$ ori;
- determinăm toate fracțiile A / B din intervalul X/N ... $(X + 1) / N$. Pentru un B fixat există cel mult un A care să corespundă; acesta poate fi determinat în $O(1)$;
- sortăm aceste fracții în $O(N \cdot \log N)$ și determinăm rezultatul.

Cum construim funcția descrisă mai sus?

Fie $C[j]$ numărul de fracții ireductibile i/j mai mici decât X/N.

Algoritmul este bazat pe următoarea observație:

$$C[j] = \lfloor X \cdot B / N \rfloor - \sum_{D \mid j} C[D].$$

O implementare directă, care testează fiecare D dacă este divizor al lui j, are ordinul de complexitatea $O(N^2)$.

O implementare mai bună, inspirată din *ciurul lui Eratostene*, este următoarea: la pasul j, cunoscând $C[j]$, îl scădem din toți multiplii lui j. Timpul de execuție al funcției este $O(N \cdot \log N)$.

Analiza complexității

Ordinul de complexitate al operațiilor de citire a datelor de intrare și scriere a rezultatelor este $O(1)$.

Operația de determinare a numărului X are ordinul de complexitate $O(N \cdot \log^2 N)$.

Pasul de după determinarea lui X are ordinul de complexitate $O(N \cdot \log N)$, dat de sortare.

În concluzie, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(N \cdot \log^2 N)$.

P060314: Traversare

Problema se rezolvă prin metoda *programarii dinamice*.

Toate calculele pentru numărul de posibilități de traversare se vor face modulo 9901.

Trebuie calculat, pentru fiecare cutie i, $A[i]$ = numărul de posibilități de a traversa lacul, ultima cutie fiind i.

Metoda clasică, folosind $j = 1, 2, ..., i - 1$ ca penultima cutie (dacă diferența de înălțime o permite) și calculând $A[i]$ pe baza valorilor din $A[j]$, ar avea ordinul de complexitate $O(N^2)$.

Există mai multe metode de complexitate $O(N \cdot \log N)$. Vom prezenta una dintre ele.

Inițial sortăm înălțimile cutiilor.

Construim o structură de date sub forma unui arbore binar complet.

Frunzele arborelui corespund înălțimilor cutiilor. A K-a frunză corespunde celei de-a K-a înălțimi ($1 \le K \le N$) și va conține numărul de posibilități (modulo 9901) în care se poate realiza o traversare, terminând cu o cutie care are această înălțime.

Nodurile intermediare din arbore conțin suma valorilor din cei doi fii modulo 9901.

Inițial considerăm că s-au folosit 0 cutii, deci arborele este plin cu 0.

La introducerea unei cutii de înălțime X se realizează următoarele operații:

- se caută binar înălțimile $X - H$ și $X + H$ în vectorul sortat de înălțimi;
- se calculează, în $O(\log N)$, numărul de posibilități în care se poate realiza o traversare, terminând cu cutii având înălțimea între $X - H$ și $X + H$. Lăsăm detalierea acestui

pas ca exercițiu pentru cititor; având explicată structura de date, exercițiul nu este foarte dificil.

- se adună 1 la acest număr, datorită posibilității de a folosi cutia nouă ca primă cutie;
- numărul de posibilități pentru înălțimea X crește cu numărul obținut și se actualizează valorile din arbore, pentru nodul tată, tatăl acestuia etc., în $\log N$ pași.

În final, din numărul total de posibilități (dat de rădăcina arborelui) se scade N, deoarece nu sunt acceptate posibilitățile de traversare cu o singură cutie.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(N)$.

Calculele pentru introducerea unei noi cutii au ordinul de complexitate $O(\log N)$. Pentru toate cele N cutii, operațiile cu arborele au ordinul de complexitate $O(N \cdot \log N)$.

Ordinul de complexitate a operației de scriere a rezultatelor este $O(1)$.

În final, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(N \cdot \log N)$.

P060315: Compania

Diferența cerută este 0 dacă N este divizibil cu 3 și 1 în caz contrar. Există mai multe metode, bazate pe *greedy* și pe *programare dinamică* cu ajutorul cărora se poate obține o soluție cu diferență minimă.

Vom prezenta soluția autorului problemei, lăsând ca exercițiu cititorului găsirea unei soluții bazate pe programarea dinamică, fără a folosi re-colorarea (prezentată mai jos). Precizăm că o astfel de soluție necesită atenție deosebită la implementare, datorită numărului mare de cazuri.

Compania este un arbore, care va fi colorat cu trei culori.

Fie un subarbore al arborelui, având ca rădăcină un nod R. Algoritmul va colora arborele respectiv în trei culori, astfel încât diferența să fie minimă. Această colorare poate fi modificată pe parcurs, după cum se va vedea în continuare.

Dacă R este o frunză, atunci va fi colorată cu culoarea 1.

Dacă R are un singur fiu, atunci se colorează subarborele care are ca rădăcină fiul respectiv, după care R este colorat cu culoarea cea mai puțin folosită, care este diferită de culoarea fiului. Dacă sunt două culori care îndeplinesc această proprietate, se va alege una oarecare. Analizând algoritmul, se poate demonstra că întotdeauna va exista cel puțin o culoare disponibilă.

Dacă R are doi fii, *FiuStânga* și *FiuDreapta*, se vor colora independent cei doi subarbori care au ca rădăcini acești fii. Apoi, arborele pornind din *FiuDreapta* este re-colorat, folosind o permutare a celor trei culori și colorarea existentă, astfel încât să se respecte condițiile din enunț și diferența în subarborele cu rădăcina R să fie minimă. R se colorează cu culoarea rămasă disponibilă.

Se observă că o astfel de re-colorare menține respectarea condițiilor în arborele cu rădăcină în *FiuDreapta*, deoarece două noduri colorate diferit rămân colorate diferit după re-colorare.

Analizând toate cazurile, se observă că o re-colorare care să ducă la o soluție validă pentru subarborele cu rădăcina în R există întotdeauna, indiferent de colorarea subarborelui cu rădăcina în *FiuStânga*.

Pentru a alege permutarea pentru re-colorare se vor încerca toate cele $3! = 6$ variante. Nu se va realiza re-colorarea efectivă, ci se va păstra, pentru *FiuDreapta*, permutarea folosită.

Algoritmul este rulat din rădăcina arborelui. Se obține astfel o strategie de colorare validă, care conduce la diferența minimă.

Presupunând că s-ar fi făcut re-colorarea efectivă la fiecare pas, culorile nodurilor ar fi fost determinate. Aceasta ar fi condus însă la o creștere a complexității, inacceptabilă pentru limitele date.

Pentru afișarea soluției se realizează o a doua parcurgere a arborelui. În cadrul acesteia se realizează colorarea efectivă, combinând permutările culorilor din fiecare nod. Pentru nodurile unde nu s-a făcut recolorare se va considera permutarea identică (1 2 3).

Analiza complexității

Operațiile de citire a datelor de intrare și scriere a rezultatelor au ordinul de complexitate $O(N)$.

La prima parcurgere, pentru fiecare nod se execută un număr de operații limitat de o constantă (cel mult două apeluri recursive și analizarea celor șase permutări posibile pentru re-colorarea subarborelui drept). Ordinul de complexitate al acestei parcurgeri este $O(N)$. Dacă s-ar fi făcut re-colorarea efectivă, se observa că unii subarbori ar fi fost re-colorați de mai multe ori, ceea ce ar fi condus la o creștere a complexității.

Cea de-a doua parcurgere are același ordin de complexitate, deoarece numărul de operații pentru fiecare nod este de asemenea limitat.

În final, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(N)$.

P060316: Consiliu de trib

Reformulăm cerința problemei: *Se consideră un arbore. Să se aleagă o ordine de colorare a nodurilor, astfel încât oricare două noduri să fie colorate diferit, fiecare nod să fie colorat cu prima culoare disponibilă și numărul de culori folosit să fie maxim. Se cere doar numărul de culori, nu și o colorare.*

Rezolvarea se bazează pe metoda *programării dinamice*.

Introducem următoarea propoziție ajutătoare:

Fie un subarbore cu o rădăcină fixată. Dacă putem colora rădăcina cu culoarea K, folosind numai nodurile din subarbore, atunci o putem colora cu orice culoare mai mică decât K.

Demonstrația este simplă și se bazează pe schimbarea ordinii de colorare a fiilor. Concluzia este că trebuie determinată, pentru fiecare nod, culoarea maximă care se poate

obține pentru acesta, folosind nodurile din subarborele care îl are ca rădăcină.

Pentru a rezolva problema se poate alege, pe rând, fiecare nod ca fiind rădăcină a arborelui (deci ca fiind ultimul nod colorat). Pentru fiecare astfel de alegere se calculează culorile maxime pentru fiecare nod, folosind un algoritm simplu de programare dinamică cu ordinul de complexitate $O(N)$. Complexitatea acestei metode este $O(N^2)$, deci metoda nu se va încadra în timp. Pe de altă parte, dacă se pornește dintr-un singur nod, se poate demonstra că soluția este cu cel mult 1 mai slabă decât soluția optimă; pornind din mai multe noduri, fără a ieși din timp, șansele de a obține soluția optimă cresc.

Pentru un nod oarecare K, considerat ca ultimul nod colorat (deci ca nouă rădăcină, reorientând arborele), ne interesează culoarea maximă cu care poate fi colorat fiecare vecin din arbore, folosind pentru aceasta numai nodurile din subarborii induși de vecinii respectivi.

Se observă că metoda precedentă ar fi dus la repetarea a numeroase calcule. Există, deci, șanse să găsim o metodă mai eficientă.

Privind problema din punct de vedere al muchiilor, ne interesează, pentru fiecare muchie, numărul maxim de culori cu care pot fi colorate capetele ei, dacă scoatem muchia respectivă și analizăm separat cei doi subarbori formați. Trebuie să calculăm două valori pentru fiecare muchie. Acestea se pot calcula folosind o procedură recursivă; se observă că fiecare muchie este parcursă de un număr constant de ori.

Pentru aflarea optimului, se analizează fiecare muchie. Dacă maximul pentru ambele capete este egal cu o valoare M, atunci arborele se poate colora cu $M + 1$ culori, alegând unul din capete ca ultim nod. Dacă maximele sunt diferite, atunci se alege ca ultim nod capătul cu maximul mai mare.

Se alege varianta cea mai bună dintre cele $N - 1$ variante date de muchiile arborelui.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(N)$, iar al operației de scriere a rezultatelor este $O(1)$.

În concluzie, ordinul de complexitate al algoritmului este $O(N)$, deoarece fiecare muchie este parcursă de un număr constant de ori.

P060317: Euro

Există mai multe rezolvări pentru această problemă.

Soluția evidentă, bazată pe metoda *programării dinamice*, are ordinul de complexitate $O(N^2)$. Aceasta constă în a calcula, pentru fiecare i între 1 și N, valoarea $C[i]$ a câștigului maxim care se poate obține în primele i zile. Acesta depinde de câștigul maxim care se poate obține în primele j zile, cu $j < i$. Această rezolvare nu s-ar încadra în timp pentru testele mari.

Să vedem cum se poate îmbunătăți această soluție.

Se observă că, dacă sunt mai multe schimbări, prima sumă schimbată este negativă. De ce? Dacă ar fi pozitivă, o putem aduna la următoarea sumă schimbată; în acest mod facem o schimbare în loc de două, economisim taxa T și obținem un profit mai mare, deoarece cursul valutar crește în timp.

Pe baza acestei observații putem împărți cele N valori în intervale. Astfel, primul interval conține primele K elemente din șir, astfel încât suma primelor K elemente să fie negativă, dar suma primelor j elemente să fie cel puțin 0 pentru orice $j < K$. Dacă un astfel de K nu există, se va efectua o singură schimbare.

Următorul interval este ales în același mod, luând în considerare numai elementele de după primul interval și așa mai departe. Se observă că toate sumele intervalelor, exceptând, eventual, ultimul interval, sunt negative; dacă una dintre sume ar fi pozitivă, am concatena intervalul respectiv cu intervalul următor, obținând o soluție mai bună.

În acest moment putem aplica metoda *programării dinamice*. Fiecare schimbare va fi efectuată exact după unul din intervale. Varianta descrisă mai sus, în care am înlocui elementele cu intervale, ar avea ordinul de complexitate $O(NR^2)$, unde NR este numărul intervalelor. Pe cazul cel mai defavorabil, se obține tot $O(N^2)$

Putem îmbunătăți această soluție.

Se observă că este preferabil ca numărul de intervale consecutive dintre două schimbări să fie mic.

Fiecare schimbare, cel mult exceptând-o pe ultima, aduce o pierdere.

Presupunem că la o anumită schimbare folosim K intervale. Dacă am "sparge" această schimbare în una formată din primele J din cele K intervale și una formată din celelalte $K - J$ intervale, ar exista o penalizare suplimentară T, dar și o scădere a pierderii, deoarece pierderea dată de primele J intervale ar fi micșorată, datorită schimbării la un curs mai mic. Pentru K suficient de mare, scăderea pierderii este semnificativ mai valoroasă decât penalizarea suplimentară T.

O limită superioară pentru K este $2 \cdot \sqrt{T}$. Dacă schimbarea ar avea mai mult de $2 \cdot \sqrt{T}$ intervale, atunci am putea-o "sparge" după $\sqrt{T}$ intervale. Aceasta ar aduce o penalizare T, dar și o scădere a pierderii. Scăderea ar fi dată de suma valorilor absolute ale pierderilor pe cele $\sqrt{T}$ intervale (egală cu cel puțin $\sqrt{T}$, deoarece pe fiecare interval pierderea este de cel puțin -1), înmulțită cu $K - \sqrt{T}$ (datorită diferenței între cursurile la care se face schimbarea). Evident, această scădere a pierderii este mai importantă decât penalizarea T.

Aceasta conduce la o îmbunătățire a algoritmului de programare dinamică. Pentru calculul $C[i]$ (câștigul după primele i intervale) se iau în considerare valorile $C[j]$, cu $j - \sqrt{T} < j < i$. Ordinul de complexitate este deci $O(NR \cdot \sqrt{T})$. Acest algoritm ar fi obținut punctajul maxim.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(N)$.

Determinarea intervalelor descrise mai sus are ordinul de complexitate $O(N)$.

Calculul valorilor $C[i]$ are ordinuul de complexitate $O(N \cdot \sqrt{T})$, deoarece numărul de intervale este limitat superior de N.

Ordinul de complexitate al operației de scriere a rezultatelor este $O(1)$.

În final, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(N \cdot \sqrt{T})$.

P060318: Nava spațială

Aceasta a fost cea mai simplă problemă din concurs.

Pentru rezolvare, putem porni invers, de la ultimul zbor spre primul.

O echipă de N oameni poate să conțină cel mult unul din ultimii S oameni de știință care au părăsit Pământul, cel mult doi din ultimii $2 \cdot S$ oameni etc.

Construim o matrice A, unde $A[i, j]$ reprezintă numărul de echipe posibile de j oameni de știință care au fost aduse cu ultimele i zboruri.

Evident, $A[1, 1] = S$ și $A[i, j] = 0$ pentru $j > i$.

Pentru $i > 1$, să analizăm cum se poate forma o echipă cu j oameni după i zboruri. O astfel de echipă poate conține cel mult P oameni din cel mai puțin recent zbor, cărora li se vor alătura $j - P$ oameni din celelalte zboruri. Evident $P \le j$ și $P \le S$.

Cei P oameni se pot alege în C_S^P moduri. Pentru fiecare astfel de alegere, ceilalți $j - P$ oameni se pot alege în $A[i - 1, j - P]$ moduri. Deci pentru P fixat avem $C_S^P \cdot A[i - 1, j - P]$ moduri.

Modalitățile pentru $P = 0$, $P = 1$, $P = 2$ etc. se adună, obținându-se $A[i, j]$. Se folosesc calcule cu numere mari.

Deoarece sunt maxim 400 de posibilități de date de intrare, valorile pentru fiecare intrare posibilă pot fi scrise într-o tabelă de constante, într-un alt program. Se obține astfel un program care citește datele de intrare și afișează direct rezultatul.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare și cel al operației de scriere a rezultatelor este $O(1)$.

Programul dat spre evaluare constă în citirea datelor și afișarea rezultatelor. Deoarece numărul de cifre al rezultatului este limitat superior de o constantă, ordinul de complexitate al rezolvării este $O(1)$.

De-ale programatorilor...

Orice bug suficient de avansat nu poate fi deosebit de o facilitate.

Dacă un program este folositor, atunci va trebui modificat cu siguranță.

Dacă un program este complet inutil, atunci va trebui documentat.

Programatorii adevărați nu își comentează niciodată codul sursă. Datorită faptului că a fost depus un efort deosebit pentru a scrie codul, este normal să trebuiască un efort și mai mare pentru a-l înțelege.

Hardware-ul este un produs care se strică dacă te "joci" mult cu el. Software-ul este un produs care funcționează numai după ce te "joci" mult cu el.

Viața ar fi mult mai ușoară dacă am putea să ne uităm peste codul sursă.

Cea mai bună modalitate de a mări viteza unui calculator este folosirea unei accelerații de 9,8 m/s^2.

Orice femeie este de nota 10. Diferența o face baza în care este scris acest număr.

Acronime

MS-DOS = Maybe SomeDay an Operating System (poate, într-o zi, un sistem de operare)

MACINTOSH = Machine Always Crashes - If Not The Operating System Hangs (mașina se blochează întotdeauna, dacă nu, cade sistemul de operare)

IBM = It's Better Manually (este mai bine de mână)

MIPS = Meaningless Indicator of Processor Speed (indicator fără sens al vitezei procesorului)

Depanare în C

- dacă este posibil, nici măcar nu încerca; lasă pe altcineva
- introdu linii vide în poziții aleatoare, recompilează și execută
- aruncă cu apă sfințită pe tastatură
- sună la 112 și țipă
- rescrie programul în limbaj de asamblare; aceasta nu va rezolva problemele dar, cel puțin, vei fi sigur că nimeni altcineva nu le va descoperi și nu te va putea face nimeni să te simți prost

22-30 noiembrie 2003, București

Concursul Național de Programare "STELELE INFORMATICII"

Mihai Stroe

Vă prezentăm în continuare soluțiile problemelor propuse spre rezolvare la Concursul național de programare "Stelele informaticii". Aceste soluții au fost realizate de redacția GInfo pe baza soluțiilor oficiale prezentate de către autorii problemelor.

Clasele a IX-a și a X-a

P080301: Cuvinte

Problema se rezolvă folosind metoda *programării dinamice*.

Pornind de la sfârșitul șirului de numere, vom construi un tablou A, în care A_i reprezintă numărul de subșiruri strict crescătoare de lungime maximă care încep cu elementul din poziția i. Acest tablou se construiește folosind rezolvarea problemei subșirului strict crescător de lungime maximă, deci se va construi în paralel un tablou L, unde L_i reprezintă lungimea celui mai lung subșir strict crescător care începe pe poziția i.

Fie MAX valoarea maximă din L.

Suma elementelor din A, corespunzătoare elementelor maxime din L, reprezintă numărul total de subșiruri strict crescătoare de lungime maximă. Ne interesează al K-lea astfel de subșir.

Fie i_1 poziția primei apariții a lui MAX în L, i_2 poziția celei de-a doua apariții ș.a.m.d. Dacă $A_{i_1} \geq K$, atunci subșirul căutat este al K-lea subșir strict crescător maximal care începe pe poziția i_1. Dacă $A_{i_1} < K \leq A_{i_1} + A_{i_2}$, atunci subșirul căutat începe cu poziția i_2 etc.

După ce a fost găsit primul element al subșirului căutat, se determină al doilea element, apoi al treilea etc. Metoda este similară. Se impune următoarea observație: dacă, de exemplu, subșirul începe cu poziția i_3, se caută al $(K - A_{i_1} + A_{i_2})$ - lea subșir crescător maximal care începe după poziția i_3, are lungimea MAX - 1, iar primul element al său este mai mare decât elementul de pe poziția i_3. Acest subșir se caută folosind același procedeu.

Analiza complexității

Operațiile de citire a datelor de intrare și scriere a rezultatelor au ordinul de complexitate $O(N)$.

Calculul elementelor tablourilor A și L are ordinul de complexitate $O(N^2)$; acesta este ordinul de complexitate al rezolvării clasice a problemei subșirului crescător de lungime maximă. Datorită limitelor din enunț și simplității implementării, acest algoritm este cea mai bună alegere în timp de concurs.

Recomandăm cititorului să încerce adaptarea algoritmului de complexitate $O(N \cdot \log N)$ pentru această problemă.

Determinarea celui de-al K-lea subșir pe baza tablourilor A și L are ordinul de complexitate $O(N)$.

În final, ordinul de complexitate al rezolvării pentru această problemă este $O(N^2)$.

P080302: Timp

Pentru a rezolva această problemă pornim de la următoarea observație: la un moment dat, în clepsidră se află cel mult două cantități nenule de nisip, X și Y.

Perechea (X, Y) poate fi tratată similar cu perechea (Y, X). Considerăm $X \leq Y$.

Din X și Y se va obține fie perechea $(X/2, Y + X/2)$, fie $(Y/2, X+Y/2)$, în funcție de sensul în care rotim clepsidra. Al doilea element este mai mare sau egal cu primul.

Considerând invers, din ce pereche se obține perechea (A, B) cu $B \geq A$?

Simplu: din (X, Y) se obține $(X/2, Y + X/2)$ sau $(Y/2, X + Y/2)$, deci elementul mai mic este unul din elementele din (X, Y) împărțit la 2.

Atunci un element din perechea care a condus la (A, B) este $2 \cdot A$ (cel mai mic element din (A, B) înmulțit cu 2). Al doilea element este $A + B - 2 \cdot A$, adică $B - A$.

Această observație conduce la următorul algoritm:

- se pleacă de la configurația finală a clepsidrei;
- se efectuează mutările inverse, ținând cont de regula "(A, B) cu $A \leq B$ provine din $(2 \cdot A, B - A)$"; rotirea spre stânga, respectiv spre dreapta sunt date de poziția elementului mai mic din pereche;
- se reconstituie șirul de mutări pornind de la configurația inițială și se afișează.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(1)$.

Se observă că după primul pas, cele două numere vor fi obligatoriu pare, după al doilea pas vor fi divizibile cu 4, iar după al K-lea pas, cu 2^K. Deoarece suma lor este N, numărul de pași este $O(\log N)$.

Ordinul de complexitate al operației de scriere a rezultatelor este $O(\log N)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(\log N)$.

P080303: Vecini

Pentru a rezolva această problemă, la început, vom genera primele 8 linii:

```
1
1 1
1 0 1
1 1 1 1
1 0 0 0 1
1 1 0 0 1 1
1 0 1 0 1 0 1
1 1 1 1 1 1 1 1
```

Completând toate liniile cu zerouri până la cea mai apropiată putere a lui 2, se obține următorul tabel:

```
1
1 1
1 0 1 0
1 1 1 1
1 0 0 0 1 0 0 0
1 1 0 0 1 1 0 0
1 0 1 0 1 0 1 0
1 1 1 1 1 1 1 1
```

Se observă că jumătatea a doua a fiecărei linii este identică cu prima jumătate. Mai mult, elementele din prima jumătate a unui rând cu 2^K elemente (după completare) sunt identice cu elementele aflate cu 2^{K-1} rânduri mai sus (dacă am completa și rândurile respective cu suficiente zerouri).

Pentru a valida aceste observații se poate demonstra prin inducție că observațiile sunt valabile în tot tabelul; concurenții pot alege o altă variantă de validare generând mai multe linii din tabel prin metoda forței brute și testând această ipoteză.

În acest moment, ideea de rezolvare devine destul de clară:

- pentru a genera linia N se generează și se dublează "prima jumătate". Am pus "prima jumătate" între ghilimele deoarece este vorba despre jumătatea după completarea până la o putere a lui 2;
- pentru a genera "prima jumătate" se apelează recursiv aceeași metodă, având ca parametru un nou N, care se obține scăzând din vechiul N cea mai mare putere a lui 2 mai mică decât el.

Există și alte variante de rezolvare eficiente, toate exploatând proprietățile descrise mai sus.

Analiza complexității

Aproximăm superior pe N cu cea mai mică putere a lui 2 mai mare sau egală cu N. Fie aceasta P.

Conceptual, sunt generate $O(P)$ elemente (P pe linia N, $P / 2$ cu $P / 2$ linii mai sus etc., suma fiind aproximativ $2 \cdot P$). Practic, se poate folosi un singur vector.

Pentru fiecare element se fac calcule în $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(N)$.

P080304: Editor

Se citesc caracterele unul câte unul și se consideră o stivă, inițial goală. De fiecare dată când se întâlnește o paranteză (deschisă sau închisă, dreaptă sau rotundă), aceasta se introduce în stivă. Dacă se întâlnește caracterul "`*`", atunci se scoate caracterul din vârful stivei (dacă există un astfel de caracter). În final (la întâlnirea caracterului "`E`"), stiva va conține șirul afișat pe ecran.

Pentru a verifica dacă un șir este parantezat corect, acesta se parcurge de la stânga la dreapta și se consideră o a doua stivă, inițial vidă. Dacă se întâlnește o paranteză deschisă dreaptă sau rotundă, aceasta se introduce în vârful stivei. Dacă se întâlnește o paranteză închisă și caracterul din vârful stivei nu este o paranteză deschisă de același tip cu paranteza închisă (dreaptă sau rotundă), atunci se semnalează eroare. Altfel, paranteza deschisă din vârful stivei se elimină și se trece la următorul caracter. Dacă nu s-a semnalat eroare în timpul parcurgerii șirului și la final stiva este vidă, atunci șirul este parantezat corect.

Analiza complexității

Notăm cu N numărul total de caractere și cu T numărul total de linii prezente în fișierul de intrare. În aceste condiții, operația de citire a datelor are un ordin de complexitate

$O(N)$, iar operația de scriere a rezultatelor are ordinul de complexitate $O(T)$.

Fiecare din cei doi pași ai rezolvării are ordinul de complexitate $O(N)$, deoarece pentru fiecare caracter din șir se execută o secvență de operații de complexitate $O(1)$.

În concluzie, ordinul total de complexitate al algoritmului de rezolvare a acestei probleme este $O(N)$.

P080305: Numere

Practic, fiecare mutare duce la schimbarea parității vecinilor poziției în care s-a efectuat mutarea, respectiv la schimbarea / menținerea parității poziției în care s-a efectuat mutarea, în funcție de numărul de vecini.

În continuare vom lucra, pentru fiecare poziție, cu restul valorii corespunzătoare la împărțirea cu 2. Problema cere să aducem toate aceste resturi fie la valoarea 0, fie la valoarea 1 și să alegem varianta cu număr minim de mutări. Vom trata primul caz, cel de-al doilea rezolvându-se în mod similar.

Se observa că M și N sunt mai mici sau egale cu 12. De asemenea, o a doua mutare într-o anumită poziție anulează efectul primei mutări în poziția respectivă, ceea ce înseamnă că în fiecare poziție se va face cel mult o mutare.

Observațiile de mai sus conduc la următorul algoritm de rezolvare:

- Se generează toate cele maxim 2^N modalități de a efectua mutări în prima linie (se mută / nu se mută în prima poziție, a doua etc.). Am scris maxim 2^N modalități deoarece în anumite poziții nu se pot efectua mutări.
- Pentru fiecare astfel de modalitate, se examinează, pe rând, necesitatea de a muta în pozițiile din rândul 2, 3, ..., M. Dacă suntem în poziția (i, j) și elementul de pe poziția $(i - 1, j)$ nu este adus la valoarea 0, este obligatoriu să mutăm în (i, j). În caz contrar este interzis să mutăm în (i, j), deoarece nu putem afecta elementul $(i - 1, j)$ cu alte mutări în linia i sau mai jos.

Sunt efectuate mutările obligatorii, ceea ce conduce la modificarea elementelor din matrice. Dacă o mutare obligatorie este împiedicată de faptul că în poziția respectivă nu se pot efectua mutări, înseamnă că varianta curentă (dată de mutările de pe prima linie) nu conduce la soluție.

După examinarea obligativității mutărilor în linia M se verifică dacă toate elementele din linia M au valoarea 0. Dacă da, varianta curentă a condus la soluție, deci se verifică dacă aceasta este mai bună decât soluția optimă obținută anterior.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(M \cdot N)$.

Sunt maxim 2^N variante de a muta pe prima linie; pentru fiecare astfel de variantă, verificarea obligativității și efectuarea mutărilor se realizează în $O(M \cdot N)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(M \cdot N)$.

P080306: Oo

Se construiesc trei tablouri unidimensionale (conceptual; practic, se va folosi unul singur):

- A_1 – elementul de pe poziția i reprezintă numărul maxim de ouă adunate până în sectorul i, dacă se adună ouăle din primele două sectoare, luate împreună;
- A_2 – elementul de pe poziția i reprezintă numărul maxim de ouă adunate până în sectorul i, dacă se adună ouăle din sectorul al doilea, împreună cu cele din al treilea;
- A_3 – elementul de pe poziția i reprezintă numărul maxim de ouă adunate până în sectorul i, dacă se adună ouăle din primul și ultimul sector, luate împreună.

Evident, în primul caz nu se pot lua ouăle din ultimul sector etc.

Valorile din aceste tablouri se pot calcula pe baza unor recurențe simple, ceea ce conduce la un algoritm liniar.

În final se selectează maximul dintre aceste valori, determinate corect.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(N)$, iar cel al operației de scriere a rezultatelor este $O(1)$.

Calculul valorilor din tablouri are ordinul de complexitate $O(N)$.

În concluzie, ordinul total de complexitate al algoritmului de rezolvare a acestei probleme este $O(N)$.

Clasele a XI-a și a XII-a

P080307: Arbori

Problema se rezolvă prin metoda *programării dinamice*.

Construim o matrice A cu M linii și $K + 1$ coloane (numerotate de la 0 la K), unde $A_{i,j}$ reprezintă efortul total minim necesar pentru a obține un subarbore pornind din nodul i al primului arbore, izomorf cu un subarbore pornind din nodul corespunzător al celui de-al doilea arbore, care să conțină exact j frunze.

Efortul este dat de formula $1000 \cdot nrCrengiTaiate + nrFloriRupte$. În acest mod se elimină `if`-uri complexe de genul: "dacă numărul de crengi este mai mic decât optimul până la momentul respectiv, sau este egal, dar numărul de flori este mai mic".

O parte din elemente sunt egale cu $+\infty$ (o valoare maximă fixată); pentru acestea nu se pot obține subarbori izomorfi cu j noduri.

Pentru a obține un subarbore cu j noduri, pornind din nodul i, avem următoarele opțiuni:

- lăsam f flori în nodul i (trebuie să putem lăsa f flori și în nodul izomorf cu i din celălalt subarbore); f este contorul într-un ciclu `for`;
- în continuare, fie nu păstrăm nici o creangă din i (deci tăiem între 0 și 4 crengi, deoarece vom tăia numai crengi care există, din primul și din cel de-al doilea arbore), fie păstrăm numai creanga stângă (dacă există în ambii arbori), fie

numai pe cea dreaptă (dacă există), fie pe amândouă (dacă există în ambii arbori);
- pentru ultimul caz, din cele $j - f$ flori, o parte (st) sunt în stânga, iar $j - f - st$ în dreapta.

Nu am descris toate condițiile care se pun pentru a nu complica prezentarea.

Elementele matricei se calculează de la frunze spre rădăcină. Reconstituirea soluției se poate face fie pe baza valorilor din matrice, fie stocând informații suplimentare.

Se pot face numeroase optimizări. De exemplu, pentru fiecare nod i se poate determina numărul maxim de flori care poate fi obținut într-un subarbore cu rădăcina i, astfel încât să existe subarborele izomorf corespunzător în cel de-al doilea arbore. Aceste valori influențează limitele `for`-urilor care apar la calculul elementelor.

Implementarea și tratarea tuturor cazurilor necesită o atenție mărită; totuși, ideea de rezolvare nu este foarte complicată, deci problema se poate rezolva în mai puțin de 90 de minute, incluzând testarea și depanarea.

Analiza complexității

Operația de citire a datelor de intrare are ordinul de complexitate $O(M + N)$.

Calculul unuia dintre cele maxim $M \cdot K$ elemente ale matricei are ordinul de complexitate $O(11 \cdot K) = O(K)$. Aici, 11 vine de la numărul de flori pe care îl lăsăm în nodul respectiv, iar K de la posibilitatea de a păstra atât subarborele stâng, cât și subarborele drept, ceea ce duce la împărțirea florilor între cei doi subarbori.

În concluzie, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(M \cdot K^2)$.

În practică, numărul fix de operații de la fiecare pas este destul de mare (apar foarte multe condiții), dar nu toți cei $M \cdot K^2$ pași amintiți sunt efectuați. De exemplu, pentru nodurile care au un singur descendent se execută $O(11)$ operații în loc de $O(11 \cdot K)$. În orice caz, algoritmul se încadrează în timp.

P080308: Sum

Problema admite o rezolvare de complexitate $O(N)$, dar și alte rezolvări de complexitate $O(N \cdot \log N)$ care merită menționate, deoarece pot fi folosite în alte probleme.

Prezentăm mai întâi rezolvarea liniară.

Construim un tablou P cu semnificația că P_i = *poziția de sfârșit a celei mai scurte secvențe, care începe în i și are suma elementelor strict pozitivă.*

Construcția acestui tablou se face de la dreapta spre stânga, folosind elementele tocmai calculate. Secvența (i, P_i) se poate numi atomică.

Se observă că dacă avem o secvență curentă care se termină în poziția i și vrem să mărim această secvență spre dreapta (cu scopul de a crește suma elementelor) atunci trebuie să sărim direct în P_i, pentru că altfel i-am scădea suma (deoarece am adăuga o cantitate negativă). Se încearcă într-un fel împărțirea șirului inițial în secvențe strict pozitive, însă această împărțire nu se face numai asupra șirului inițial, ci a oricărei secvențe.

Deși aparent construcția acestui tablou are ordinul de complexitate $O(N^2)$, o analiză amortizată ne arată că este vorba de $O(N)$.

Cu ajutorul tabloului P, ne "plimbăm" cu doi *pointer*-i (indici) în șirul inițial (limita stângă și limita dreaptă a secvenței căutate, inițial plecând din stânga șirului), având grijă ca diferența dintre cei doi să fie între L și U. În căutarea secvenței *pointer*-ii sar numai peste secvențe atomice (din i se sare în P_i). Fiecare secvență căutată se compară cu maximul până în acel moment. Astfel și această parte de căutare are ordinul de complexitate $O(N)$.

Recomandăm cititorului să analizeze comportamentul algoritmului pe un exemplu.

Vom prezenta în continuare două variante de rezolvare care au ordinul de complexitate $O(N \cdot \log N)$.

Varianta cu heap-uri

Se folosește un *max-heap* cu $U - L + 1$ elemente. La fiecare pas i, în acest *heap* se află sumele secvențelor de elemente din șir, cu lungimea cuprinsă între L și U, care se termină pe poziția i.

De fapt, în *heap* nu se memorează efectiv sumele, ci pozițiile de început ale acestor secvențe, astfel încât sumele să se afle în relația impusă de *max-heap*.

Avansarea la pasul $i + 1$ are următorul efect:
- toate secvențele din *heap* devin mai lungi cu un element. Aceasta nu modifică relația de *max-heap* dintre ele, pentru că la fiecare se adaugă aceeași cantitate;
- cea mai lungă secvență va avea lungimea $U + 1$ (dacă $i > U$), deci va trebui scoasă din *heap*;
- apare o nouă secvență de lungime L, care va trebui introdusă în *heap*.

Suma unei secvențe (cunoscând pozițiile de început și sfârșit) poate fi calculată folosind vectorul sumelor parțiale. Dacă șirul dat este A, atunci definim $S_i = A_1 + A_2 + ... + A_i$, $S_0 = 0$. Atunci, suma elementelor dintre pozitiile i și j este $S_j - S_{i-1}$.

Pentru localizarea secvențelor care trebuie scoase din *heap* se păstrează o informație auxiliară. De asemenea, se observă că va fi scos un element aflat pe o poziție oarecare, nu neapărat din vârful *heap*-ului.

Evident, la fiecare pas maximul este comparat cu soluția optimă de până atunci, și eventual aceasta este actualizată.

Recomandăm implementarea acestei soluții, ca un exercițiu util pentru lucrul cu *heap*-uri.

La fiecare pas, se fac cel mult două operații cu *heap*-uri, care pot fi realizate cu complexitate *logaritmică*. Ordinul de complexitate al algoritmului este $O(N \cdot \log (U - L))$, adică $O(N \cdot \log N)$ pentru cazul cel mai defavorabil.

Varianta bazată pe programare dinamică

Se calculează, pentru fiecare poziție i, cea mai mare sumă a unei secvențe de lungime cel mult $1, 2, 4, ... 2^K$ începând din

poziția respectivă. Fie această informație $MAX[K, i]$. Notăm cu $LUNG[K, i]$ lungimea acestei secvențe.

Inițial $MAX[0, i]$ este elementul de pe poziția i din șir.

La trecerea de la pasul K la pasul $K + 1$, secvența de sumă maximă, de lungime cel mult 2^{K+1}, care începe de pe poziția i, este fie secvența de lungime cel mult 2^K (cu suma $MAX[K, i]$), fie o secvență mai lungă. Singura secvență candidată este cea formată din cele 2^K elemente începând de pe poziția i, la care alipim încă $LUNG[K, i + 2^K]$ elemente care dau cea mai mare sumă a unei secvențe imediat următoare, de lungime maxim 2^K.

Calculul valorilor din MAX și $LUNG$ pentru un K fixat are deci ordinul de complexitate $O(N)$, ceea ce înseamnă că determinarea tuturor valorilor va avea ordinul de complexitate $O(N \cdot \log N)$.

Cum folosim aceste valori?

Se observă că o secvență de lungime cel puțin L și cel mult U începe cu o secvență de lungime L, la care se alipesc la dreapta cel mult $U - L$ elemente. Fie P cea mai mare putere a lui 2 mai mică decât $U - L$.

Atunci, secvența de sumă maximă, de lungime cel puțin U și cel mult L, care începe cu L elemente din care ultimul se află pe poziția $i - 1$, se încadrează în unul din următoarele cazuri:

- are exact lungimea L;
- are lungimea cuprinsă între L și $L + P$, caz în care se adaugă la cele L elemente o secvență de lungime $LUNG[P, i]$;
- are lungimea mai mare de $L + P$. Fie $D = U - L$. Atunci secvența se termină pe una din ultimele P poziții din cele U, deci se adaugă $D - P$ elemente la primele i, după care urmează încă $LUNG[P, i + D - P]$ elemente.

Se ia varianta cea mai bună din aceste cazuri. Calculele necesare au ordinul total de complexitate $O(N)$.

Se impun următoarele observații:

- la fel ca la varianta cu *heap*-uri, se va folosi vectorul sumelor parțiale;
- se poate renunța la valorile din MAX; se lucrează numai cu $LUNG$ și sumele parțiale;
- din matricea $LUNG$ se vor folosi numai ultimele două linii; linia $K + 1$ se construiește pe baza liniei K, după care se renunță la linia K;
- pentru a rezolva problema, indicele K al pasului pentru calcularea valorilor din $LUNG$ merge până la P, deci sunt $\log (U - L)$ pași și nu $\log N$ pași (valorile care ar fi obținute la pașii următori nu ne interesează).

În concluzie, ordinul de complexitate al algoritmului este $O(N \cdot \log (U - L))$, adică $O(N \cdot \log N)$ pentru cazul cel mai defavorabil.

P080309: Taxi

Această problemă nu este dificilă; dificultatea ei constă în a nu pierde din vedere vreun caz.

Notăm cu dx diferența pe x între cele două taxiuri și cu dy diferența pe y.

Avem următoarele cazuri:

- $dx = 0$, $dy = 0$ (taxiurile coincid): toți vor fi nehotărâți;
- $dx > dy$: sunt $B + 1$ oameni nehotărâți;
- $dx < dy$: sunt $A + 1$ oameni nehotărâți;
- $dx = dy$: aici este un caz special. Pe diagonala dintre taxiuri vor fi oameni nehotărâți, însă vor exista, de asemenea, și două dreptunghiuri în care toți oamenii sunt nehotărâți. Să presupunem că taxiurile sunt colțurile unui pătrat. Dreptunghiurile (cu oameni nehotărâți) vor avea un colț în colțurile neocupate (de taxiuri) ale pătratului, iar altul în colțul orașului.

Analiza complexității

Ordinul de complexitate al operațiilor de citire a datelor și scriere a rezultatelor este $O(1)$.

Ordinul de complexitate al rezolvării unui caz de test este $O(1)$, deoarece identificarea cazurilor și calculele aferente necesită un număr de operații mic, limitat superior de o constantă.

P080310: Expr

După citirea datelor, se sortează toate numerele din mulțimi. Pe parcursul rezolvării, un element va fi înlocuit prin indicele său din vectorul sortat; pentru aceasta se folosește căutarea binară.

Rezolvarea problemei se bazează pe evaluarea expresiilor cu ajutorul gramaticilor. Gramatica este structurată pe două niveluri: *expresie* și *operand*. Expresia este formată din mai mulți *operanzi* între care există operațiile specificate, iar operandul poate fi o mulțime sau altă expresie (dacă se întâlnește paranteza deschisă).

Pentru fiecare dintre cele două elemente ale gramaticii se va scrie o funcție recursivă. Aceasta returnează o mulțime, reprezentată printr-o structură de 1000 de octeți, suficienți pentru totalul de 8000 de numere distincte din expresie (din considerente de eficiență se poate prefera folosirea unei structuri de 1024 octeți). Această structură poate fi alcătuită din elemente de un octet (tipul *byte* în *Pascal*); se pot alege variante pe doi, respectiv patru octeți.

Pentru mai multe detalii vom analiza exemplul din enunț: `{1,2,3,4}%({1,2,3,4}*{}+{1,2}+{5,6}-{1})`

Inițial se apelează funcția *expresie*().

Expresia conține un operand, urmată eventual de un operator și de o altă expresie. Se va apela deci funcția *operand*(), se va citi un operator (dacă nu s-a terminat șirul), apoi se va apela funcția *expresie*(). Rezultatul apelării ei și rezultatul apelării funcției *operand*() vor servi ca operanzi operatorului respectiv.

Operandul inițial este dat de mulțimea `{1,2,3,4}`. Se citește apoi `%` și se apelează *expresie*().

Aceasta întâlnește un prim operand dat de paranteza deschisă, deci se va apela din nou *expresie*(), după care se va citi o paranteză închisă. Această nouă apelare va întoarce rezultatul `{1,2,3,4}*{}+{1,2}+{5,6}-{1}` (evident, prin noi apelări de *operand*() și *expresie*()). Fișierul se ter-

mină, deci se iese din funcțiile recursive și se calculează și celelalte rezultate.

Invităm cititorul să realizeze o implementare a acestei rezolvări. Este un exercițiu util de programare, testare și depanare (dacă este cazul).

Precizăm că, în cazul în care s-ar fi introdus priorități ale operatorilor, gramatica ar fi fost mai complicată.

Analiza complexității

Fie L numărul de caractere din expresie, N numărul total de numere din mulțimile din expresie, D numărul de numere distincte și T numărul de operații.

Operația de citire a datelor de intrare are ordinul de complexitate $O(L)$.

Sortarea celor N numere are ordinul de complexitate $O(N \cdot \log N)$.

Pe parcursul rezolvării, fiecare număr este transformat în indicele său din vectorul sortat, ceea ce duce la ordinul de complexitate cu $O(N \cdot \log D)$. Deoarece $D \leq N$, acest termen poate fi ignorat.

Efectuarea fiecăreia din cele T operații necesită $O(D)$ operații elementare. Pentru acestea se folosesc operațiile *OR*, *AND*, *XOR* pe octeții care compun structura.

În concluzie, ordinul de complexitate al algoritmului este $O(L + N \cdot \log N + T \cdot D)$. Relația pare complicată, dar ea ne spune că rezolvarea se va încadra în timp.

P080311: Ciclu

Rezolvarea se bazează pe o legătura subtilă dintre ciclul minim și ciclul mediu minim într-un graf care admite arce de cost pozitiv, zero și negativ. Aceste costuri au întotdeauna același semn.

Facem o altă observație: dacă adăugăm (sau scădem) aceeași valoare x la toate costurile muchiilor, costul ciclului mediu minim crește (respectiv scade) exact cu aceeași valoare x.

Se conturează deja o idee. Putem afla dacă un anumit număr x este costul cerut astfel: scădem x din costurile muchiilor și verificăm dacă ciclul minim are costul 0. Mai mult, de aici putem să aflăm dacă x este prea mic sau prea mare. Dacă x este prea mare, atunci va exista un ciclu negativ. Dacă nu, x este bun sau este prea mic.

De aici devine evidentă soluția: o căutare binară a rezultatului, în care verificăm existența unui ciclu negativ. În etapa de verificare se va folosi algoritmul *Bellman-Ford* care are ordinul de complexitate $O(N \cdot M)$. Complexitatea finală a algoritmului este $O(N \cdot M \cdot \log X)$, unde X este valoarea maximă a costului unui arc, înmulțită cu 100 (pentru cele 2 zecimale exacte).

O altă soluție care ar fi putut obține în jur de 70 de puncte, dar care merită menționată, deoarece poate fi folosită și în rezolvarea altor probleme, este bazată pe *programarea dinamică*. Această variantă are avantajul că nu lucrează decât cu numere întregi (cu excepția unor verificări).

Se alege un nod de start s și se construiește (păstrând doar ultimele două linii) matricea A cu semnificația că $A(i, j)$ este costul minim al unui drum care conține exact i arce, care începe cu nodul s și se termină cu nodul j. Evident, $A(i, j)$ se calculează folosind valorile $A(i - 1, k)$ pentru care avem arc de la k la j.

Valorile care ne interesează sunt $A(i, s)/i$; o astfel de valoare reprezintă lungimea medie a ciclului care începe în s, se termină în s și conține exact i arce.

Optimizarea majoră se bazează pe faptul că, odată ce au fost considerate ciclurile care încep dintr-un anumit nod, nu mai este nevoie ca acesta să fie considerat pe viitor. Astfel, pentru un anumit nod s, nu sunt calculate decât valorile din matrice pentru care $j \geq s$.

Complexitatea acestei rezolvări este $O(N^2 \cdot M)$; se alege, pe rând, un nod de start, după care valorile sunt construite în $O(N \cdot M)$. Acest fapt este evident dacă observăm că valorile pe o linie a matricei se construiesc în $O(M)$, deoarece valorile unui element depind de valorile asociate vecinilor din graf, aflate pe linia precedentă.

P080312: Prefix

Pentru fiecare șir se calculează funcția *prefix* corespunzătoare algoritmului *Knuth-Morris-Pratt* (*KMP*).

Se consideră că pozițiile caracterelor din șir sunt numerotate de la 1 la N (lungimea șirului). Lungimea celui mai lung prefix periodic este, dacă un astfel de prefix există, cel mai mare număr natural K ($1 < K \leq N$), având proprietățile:

- $P[K] > 0$;
- K *modulo* $(K - P[K]) = 0$.

Prin $P[K]$ s-a notat valoarea funcției prefix corespunzătoare poziției K. Dacă nu vi se pare evident că un astfel de șir este periodic, încercați să demonstrați.

Dacă prefixul de lungime K al șirului este periodic, atunci $K - P[K]$ reprezintă lungimea celei mai scurte perioade a acestui prefix.

De exemplu, pentru șirul `"abcababcababca"`, funcția prefix determinată este următoarea:

```
      a b c a b a b c a b  a  b  c  a
K:    1 2 3 4 5 6 7 8 9 10 11 12 13 14
P[K]: 0 0 0 1 2 1 2 3 4 5  6  7  8  9
```

Cel mai lung prefix periodic are lungimea 10, deoarece $P[10] = 5 > 0$ și 10 *mod* $(10 - P[10]) = 10$ *mod* $5 = 0$. De asemenea, se observă că lungimea perioadei `"abcab"` este $5 = 10 - P[10]$.

Analiza complexității

Operația de citire a datelor de intrare are ordinul de complexitate $O(N)$.

Calculul funcției prefix din cadrul algoritmului *KMP* are ordinul de complexitate $O(N)$. Pentru fiecare valoare a funcției prefix, verificarea ($P[K] > 0$ și K *modulo* $(K - P[K]) = 0$) are ordinul de complexitate constant.

În concluzie, ordinul de complexitate al algoritmului este $O(N)$.

Rundele #01 - #05

În cadrul acestui articol vom publica detalii despre desfășurarea primelor cinci runde ale ediției 2003/2004 a concursului de programare Bursele Agora organizat de revista noastră.

P080313: Cei șapte pitici

Zonele în care se află piticii pot fi considerate ca fiind nodurile unui graf orientat. Două noduri ale grafului vor fi unite printr-un arc dacă cele două zone corespunzătoare se vor afla una deasupra celeilalte.

Sensul arcului va fi dinspre zona de deasupra spre cea de sub ea. Costul arcului va fi dată de lungimea celui mai scurt puț care poate uni zonele respective. Graful este orientat, deoarece puțurile nu pot fi săpate de jos în sus.

După construirea grafului, problema se reduce la determinarea unui arbore parțial minim într-un graf orientat.

Au rezolvat corect...

La prima etapă a concursului nostru au participat peste 100 de concurenți. Dintre aceștia doar 12 au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Irinel Besliu, România
Victor Bilan, România
Ionuț-Bogdan Brîndușa, România
Adrian Cârcu, România
Dragoș Cavulea, România
Porobic Danica, Serbia și Muntenegru
Ionuț-Ovidiu Georgescu, România
Sergiu Hlihor, România
Aurelian Marinescu, România
Alexandru Popa, România
Tudor-Alexandru Rabaea, România

P080314: Asteroizi

Aceasta este o problemă clasică în care poate fi aplicat algoritmul lui *Lee*.

Vom construi o matrice care va conține numărul schimbărilor de direcție necesare ajungerii într-o anumită poziție. Rezultatul final va fi cea mai mică valoare de pe ultima linie a acestei matrice.

Pentru a construi această matrice vom marca cu 1 toate pozițiile în care se poate ajunge în momentul în care se intră în centura de asteroizi (pe orice direcție).

La următorul pas vom marca cu 2 toate pozițiile în care se poate ajunge din pozițiile marcate cu 1, dar care nu au fost marcate la pașii anteriori.

În continuare vom marca cu 3 pozițiile în care se poate ajunge din pozițiile marcate cu 2, dar care nu au fost marcate la pașii anteriori.

Se observă că, datorită faptului că numărul schimbărilor de direcție crește la fiecare pas, ne putem opri în momentul în care reușim să marcăm un element de pe ultima linie (valoarea acestuia va reprezenta numărul minim de schimbări de direcție care trebuie efectuate).

Au rezolvat corect...

Dintre concurenții care au participat la această etapă a concursului, 15 au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Liviu Ciortea, România
Adrian Diaconu, România
Marius-Dan Dragomir, România
Ivan Kazmenko, Federația Rusă
Vasanth Jai, India
Haryanto Lego, Statele Unite ale Americii
Cosmin Matei, România
Kiril Minkov, Bulgaria
Silvestru-Cosmin Negrușeri, România
Mihai-Vlad Pantiș, România
Daniel Păsăilă, România
Andrei-Ionuț Tudor, România

P080315: Recurență

Se cunoaște faptul că șirul dat de rapoartele a două numere aflate pe poziții consecutive în șirul lui *Fibonacci* are o limită care este aproximativ 1,618.

Mai mult, orice șir care este construit pe același principiu (un element este suma celor două elemente anterioare), dar pentru valorile primelor două elemente se aleg alte numere, are aceeași limită. În plus, limita este obținută foarte repede.

Pentru a rezolva această problemă, datorită faptului că se ajunge foarte repede la limită, este suficient să calculăm elementele șirului până în momentul în care valorile elementelor nu se mai modifică (folosind o precizie de opt zecimale) sau am ajuns la elementul căutat.

Au rezolvat corect...

Această problemă a fost rezolvată corect de 10 dintre participanții la cea de-a treia etapă a acestei ediții a concursului nostru:

Mugurel-Ionuț Andreica, România
Cristian Bojinovici, România
Adrian Cârcu, România
Ioan-Dan Docleanu, România
Marius Dumitran, România
Ivan Kazmenko, Federația Rusă
Andi-George Matei, România
Dalian Valeriu Moraru, România
Silvestru-Cosmin Negruşeri, România
Bogdan Nicolae, România

P080316: Parada piticilor

Problema propusă spre rezolvare la cea de-a patra etapă se reducea la eliminarea unui număr minim de elemente ale unui șir, astfel încât șirul rămas să fie palindrom (să fie identic, indiferent dacă este citit de la stânga spre dreapta sau de la dreapta spre stânga).

O modalitate de rezolvare a acestei probleme constă în determinarea celui mai lung subșir comun al șirului dat și al inversului șirului dat. Prin inversul șirului dat înțelegem șirul obținut prin considerarea elementelor în ordine inversă.

Este evident faptul că acest șir va fi palindrom și, în același timp, va fi cel mai lung astfel de palindrom. Determinarea celui mai lung subșir comun a două șiruri este o aplicație foarte cunoscută a metodei *programării dinamice* și poate fi studiată în marea majoritate a lucrărilor care tratează acest subiect.

Au rezolvat corect...

Cea de-a patra problemă de la această ediție a concursului nostru a fost rezolvată corect de 14 participanți:

Călin Bindea, România
Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Ivan Kazmenko, Federația Rusă
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Silvestru-Cosmin Negruşeri, România
Mihai-Vlad Pantiș, România
Daniel Păsăilă, România
Mihail-Cosmin Piț-Rada, România
Victor Podeanu, România

P080317: Pluto

În cazul în care considerăm că cercurile sunt vârfurile unui graf orientat, problema se reduce la alegerea unei configurații a arcelor, astfel încât fiecare nod să aibă un grad exterior dat (numărul săgeților care pleacă din cercul corespunzător) și un grad interior dat (numărul săgeților care ajung la cercul corespunzător).

Pentru aceasta vom crea o rețea de transport care va conține $2 \cdot N + 2$ noduri (N reprezintă numărul cercurilor). Două dintre acestea vor fi sursa și destinația.

N dintre noduri vor fi legate de sursă prin arce ale căror capacități vor fi egale cu gradele interioare ale nodurilor.

Celelalte N noduri vor fi legate de destinație prin arce ale căror capacități vor fi egale cu gradele exterioare ale nodurilor.

De la fiecare dintre primele N noduri vor pleca arce de capacitate 1 spre fiecare dintre celelalte N noduri.

În această rețea vom determina fluxul maxim. Dacă acesta va fi M (numărul total al săgeților care trebuie desenate), atunci săgețile pot fi desenate (graful poate fi construit).

Arcele rețelei de transport pe care există flux vor indica modul în care vor fi desenate săgețile (configurația arcelor).

Au rezolvat corect...

Această problemă s-a dovedit a fi cea mai dificilă dintre cele propuse spre rezolvare la primele cinci runde, chiar dacă ea reprezintă o aplicație clasică a algoritmilor de determinare a fluxurilor în rețelele de transport.

O primă dovadă în acest sens o constituie numărul mic al concurenților care au trimis soluții pentru această problemă. Totuși, dintre cei care au încercat să rezolve problema, șapte participanți au obținut punctajul maxim.

Mugurel-Ionuț Andreica, România
Ivan Kazmenko, Federația Rusă
Kiril Minkov, Bulgaria
Mircea Pașoi, România
Daniel Păsăilă, România
Claudiu Rad, România
Sorin Stancu-Mara, România

Un prim bilanț

Ca de obicei, o mulțime dintre cititorii revistei noastre au decis să participe la acest concurs pentru a se pregăti pentru olimpiadele care, în momentul în care citiți aceste rânduri, sunt în plină desfășurare.

După cum probabil ați văzut în regulament, numărul rundelor de la această ediție nu este specificat. Totuși, intenția noastră este de a propune spre rezolvare cincizeci de probleme la prima fază a concursului. Așadar, clasamentul publicat în acest număr reflectă situația după desfășurarea a numai o zecime dintre cele 50 de runde pe care le-am programat.

Bursele Agora 2003/2004 Top 50

1.	Mugurel-Ionuț Andreica, București	454
2.	Ivan Kazmenko, Federația Rusă	454
3.	Adrian Cârcu, Bistrița	408
4.	Kiril Minkov, Bulgaria	352
5.	Daniel Păsăilă, Suceava	316
6.	Liviu Ciortea, București	301
7.	Silvestru-Cosmin Negrușeri, Bistrița	300
8.	Lego Haryanto, Statele Unite ale Americii	281
9.	Mihai-Vlad Pantiș, Cluj-Napoca	266
10.	Mircea Pașoi, Ploiești	263
11.	Silviu-Ionuț Gănceanu, Bârlad	248
12.	Cosmin Matei, Ploiești	230
13.	Marius Dumitran, București	227
14.	Marius Dragomir, Alba Iulia	219
15.	Andrei-Ionuț Tudor, Slobozia	218
16.	Victor Podeanu, București	210
17.	Chattopadhyay Amit, India	200
18.	Vasanth Jai, India	191
19.	Tudor-Alexandru Rabaea, Constanța	183
20.	Sorin Stancu-Mara, București	181
21.	Mihail-C. Pițt-Rada, Dr. Tr. Severin	179
22.	Sergiu Hlihor, Lugoj	172
23.	Irinel Besliu, Vaslui	165
24.	Cristian Bojinovici, București	160
25.	Adrian Vladu, București	142
26.	Alexandru Popa, București	139
27.	Răzvan Mosincat, Carei	134
27.	Valentin Stanciu, București	134
29.	Călin Bindea, Bistrița	133
29.	Gabriel Kreindler, București	133
31.	Dimitrov Anton, Bulgaria	132
32.	Dalian-Valeriu Moraru, București	127
33.	Andrei Păduraru, Vaslui	125
33.	Claudiu Rad, Deva	125
35.	Ionuț-Ovidiu Georgescu, Ploiești	119
36.	Aurelian Marinescu, Brașov	116
37.	Dragoș Cavulea, Botoșani	115
38.	Alexandru Peptan, Drobeta Tr. Severin	113
39.	Ioan-Dan Docleanu, Nădlac	111
40.	Cătălin Zima, Baia Mare	110
41.	Porobic Danica, Serbia și Muntenegru	109
42.	Adrian Diaconu, București	106
43.	Ionuț-Bogdan Brîndușa, Bistrița	103
43.	George Matei, București	103
45.	Victor Bilan, Bistrița	100
45.	Bogdan Nicolae, Sibiu	100
47.	Mika Marcin, Canada	89
48.	Ciprian Păduraru, Bârlad	83
49.	Bogdan-Gabriel Ionescu, Craiova	80
50.	Cristina Stancu-Mara, București	79

Rundele #06 - #15

În continuare vă vom prezenta enunțurile celor 10 probleme propuse spre rezolvare la rundele 06-15 ale ediției din acest an a concursului de programare Bursele Agora.

P010416: Arena din Erathia

O mulțime de creaturi din *Erathia* se află în arenă. Ele se vor lupta pentru a demonstra care este cea mai puternică rasă.

Vrăjitorii din *Bracada* au teleportat creaturile în diverse poziții ale *Arenei*. Pentru a putea colabora, două creaturi din aceeași rasă pot comunica folosind tehnici ale *Școlii de Magie a Spiritului*. Totuși, calitatea comunicării telepatice depinde de distanță. Așadar, o rasă va fi dezavantajată dacă două creaturi ale rasei respective se află la distanță mare.

Se cunosc coordonatele la care au fost teleportate creaturile și trebuie determinată cea mai defavorizată rasă. Cu alte cuvinte, va trebui să determinați, pentru fiecare rasă, distanța maximă dintre doi reprezentanți ai rasei respective și apoi, pe baza distanțelor determinate, să alegeți cea mai defavorizată rasă.

Date de intrare

Fișierul de intrare **ARENA.IN** conține pe prima linie un singur număr natural N, reprezentând numărul de creaturi.

Pe fiecare dintre următoarele N linii se află trei valori x, y și r, separate prin spații. Primele două valori sunt numere reale și reprezintă coordonatele la care se află creatura, iar cea de-a treia este un șir de caractere care reprezintă rasa creaturii.

Date de ieșire

Fișierul de ieșire **ARENA.OUT** trebuie să conțină o singură linie pe care se va afla un șir de caractere reprezentând denumirea rasei celei mai defavorizate.

Restricții

- $2 \le N \le 5000$;
- coordonatele creaturilor sunt numere reale cuprinse între 0 și 1000 și conțin cel mult două zecimale exacte;
- numele creaturilor sunt formate din cel mult 25 de caractere; acestea pot fi litere mici ale alfabetului englezesc sau spații;
- vor exista cel mult 1000 de rase;
- pentru fiecare rasă există cel puțin doi reprezentanți.

Exemplu

ARENA.IN

```
7
1 1 dragon
1 2 minotaur
1 3 wood elf
1 4 wood elf
1 5 wood elf
2 5 minotaur
2 6 dragon
```

ARENA.OUT

```
dragon
```

Explicație

Distanța dintre cei doi *dragoni* este aproximativ 5.099, distanța dintre cei doi *minotauri* este aproximativ 3.162, iar distanțele dintre *elfi* sunt 1 sau 2. Așadar, *dragonii* sunt cei mai defavorizați (doar din punct de vedere al capacității de comunicare!).

Timp maxim de execuție/test: 1 secundă

P010417: Merlin

Celebrul vrăjitor *Merlin* trebuie să îndeplinească un nou ritual magic care îl va ajuta să îi înfrângă pe inamicii regelui *Arthur*. Ritualul constă în amplasarea a M pietricele într-o zonă pătratică cu latura N. Pietricelele trebuie amplasate în așa fel încât oricare patru dintre ele să nu se afle în colțurile unui pătrat cu laturile paralele cu cele ale zonei pătratice.

Date de intrare

Fișierul de intrare **MERLIN.IN** conține pe prima linie numărul natural N, reprezentând latura zonei pătratice. Cea de-a doua linie conține numărul natural M, reprezentând numărul de pietricele.

Date de ieșire

Fișierul de ieșire **MERLIN.OUT** trebuie să conțină configurația zonei pătratice. Aceasta va fi reprezentată printr-o matrice pătratică ale cărei elemente pot fi 0 sau 1. Valoarea 0 indică faptul că în locația respectivă nu se află nici o pietricică, iar valoarea 1 indică faptul că locația conține o pietricică.

Așadar, fișierul va conține N linii, pe fiecare dintre ele aflându-se câte N valori 0 sau 1. Aceste valori nu vor fi separate prin spații.

Restricții și precizări

- $3 \leq N \leq 100$;
- $1 \leq M \leq 5000$;
- într-o locație poate fi amplasată cel mult o pietricică;
- dacă există mai multe soluții, poate fi aleasă oricare dintre ele;
- va exista întotdeauna cel puțin o soluție.

Exemplu

MERLIN.IN	MERLIN.OUT
10	1111111111
54	1001010101
	1001000011
	1110000001
	1000001011
	1100000101
	1000100011
	1100110001
	1010101001
	1111111110

Timp maxim de execuție/test: 1 secundă

P010418: Distribuție

Revista *GInfo* este tipărită în două tipografii și este distribuită în N orașe. În fiecare lună jumătate dintre orașele în care este difuzată *GInfo* primesc reviste de la una dintre tipografii, iar cealaltă jumătate de la cealaltă tipografie.

De la fiecare tipografie pornesc câte $N / 2$ camioane încărcate cu reviste. Fiecare dintre cele N camioane trebuie să ajungă la unul dintre cele N orașe.

Departamentul de distribuție al *GInfo* dorește să determine care orașe vor fi deservite de prima tipografie și care de cea de-a doua, astfel încât distanța totală pe care o vor parcurge camioanele să fie minimă. Un camion va ajunge la orașul destinație pe cel mai scurt drum posibil.

Între orașe se află un număr total de M străzi pe care se poate circula în ambele sensuri.

Pentru fiecare stradă sunt cunoscute orașele pe care le unește, precum și lungimea străzii.

Orașele sunt identificate prin numere cuprinse între 1 și N, iar tipografiile se află în orașele identificate prin 1 și 2. Evident, distanța dintre tipografia din orașul 1 și orașul 1 este 0, iar distanța dintre tipografia din orașul 2 și orașul 2 este tot 0.

Date de intrare

Fișierul de intrare **GINFO.IN** conține pe prima linie numărul natural N, reprezentând numărul orașelor, precum și numărul natural M, reprezentând numărul străzilor. Cele două numere sunt separate printr-un spațiu. Fiecare dintre următoarele M linii va conține câte trei numere naturale x, y și L, cu semnificația: *există o stradă de lungime L între orașele x și y*.

Date de ieșire

Fișierul de ieșire **GINFO.OUT** trebuie să conțină două linii, pe fiecare aflându-se câte $N / 2$ numere naturale, separate prin câte un spațiu. Numerele de pe prima linie reprezintă orașele deservite de prima tipografie, iar cele de pe a doua linie reprezintă orașele deservite de cea de-a doua tipografie.

Restricții și precizări

- $1 \leq N \leq 1000$;
- $1 \leq M \leq 10000$;
- numărul orașelor este întotdeauna par;
- între oricare două orașe există cel mult o stradă;
- străzile nu se intersectează;
- dacă există mai multe soluții, poate fi aleasă oricare dintre ele;
- va exista cel puțin un drum de la fiecare dintre cele două tipografii la toate orașele;
- lungimile străzilor sunt numere naturale cuprinse între 1 și 1000.

Exemplu

GINFO.IN	GINFO.OUT
6 8	1 4 5
1 2 5	2 3 6
1 4 8	
1 6 7	
2 3 6	
3 4 5	
3 6 2	
4 5 3	
5 6 2	

Timp maxim de execuție/test: 1 secundă

P010419: Cerc

Galul *Obelix* vrea să îl convingă pe druidul *Panoramix* să îi dea puțină poțiune magică. Nemairezistând insistențelor lui *Obelix*, *Panoramix* îi promite că îi va da o picătură dacă reușește să găsească soluția unei probleme.

Panoramix așează pe un cerc N papirusuri pe care se află numerele de la 1 la N, în această ordine. Așadar, lângă numărul 1 se află, în stânga, numărul N, iar în dreapta numărul 2, lângă numărul 2 se află, în stânga, numărul 1, iar în dreapta numărul 3 etc. La fiecare pas, *Obelix* are voie să interschimbe două numere vecine. *Obelix* va trebui să efectueze un număr minim de interschimbări astfel încât numărul 1 să ajungă în locul numărului N, numărul 2 să ajungă în locul numărului N - 1 etc.

Datorită faptului că *Obelix* nu stă prea bine la capitolul "inteligență", el cere ajutorul prietenului său *Asterix* care, după multe proteste, acceptă să îl ajute. Cum magia este la ordinea zilei, tocmai ați intrat în pielea unui spion imperial care trebuie să știe câte interschimbări va face *Asterix*.

Date de intrare
Fișierul de intrare **CIRCLE.IN** conține o singură linie pe care se va afla numărul natural N.

Date de ieșire
Fișierul de ieșire **CIRCLE.OUT** trebuie să conțină o singură linie pe care se va afla numărul minim de interschimbări care trebuie efectuate.

Restricție
- $3 \le N \le 30000$.

Exemplu

CIRCLE.IN	CIRCLE.OUT
4	2

Explicație
La primul pas vor fi interschimbate numerele 1 și 4, iar la al doilea vor fi interschimbate numerele 2 și 3.

Timp maxim de execuție/test: 1 secundă

P010420: Explozie
Flota romulană asediază planeta *Vulcan* cu scopul declarat de a alipi spațiul vulcanian celui controlat de *Imperiul Romulan*. Un număr inimaginabil de navete cu soldați romulani au intrat în sistemul stelar vulcanian. Vulcanienii au o singură armă la dispoziție, care poate fi detonată în orice punct al spațiului (nu prea contează unde, deoarece romulanii ocupă, practic întregul sistem).

Vom considera că navetele romulane ocupă fiecare punct de coordonate întregi a spațiului. Arma vulcaniană va produce o explozie care va distruge toate navele aflate în interiorul razei de acțiune a armei. Așadar, va fi afectată o zonă sferică a spațiului. Vor fi distruse toate navele, inclusiv cele aflate pe suprafața acestei sfere.

Cunoscând raza de acțiune R, va trebui să determinați numărul navelor romulane distruse.

Date de intrare
Fișierul de intrare **WEAPON.IN** conține o singură linie pe care se va afla numărul natural R, reprezentând raza de acțiune a armei.

Date de ieșire
Fișierul de ieșire **WEAPON.OUT** trebuie să conțină pe o singură linie numărul navelor romulane care vor fi distruse.

Restricție
- $1 \le N \le 10000$.

Exemplu

WEAPON.IN	WEAPON.OUT
1	7

Timp maxim de execuție/test: 1 secundă

P010421: Recunoașterea caracterelor
Se consideră o matrice pătratică cu 16 linii și 16 coloane. Elementele acestei matrice pot fi 0 sau 1. Această matrice reprezintă, de fapt, o literă a alfabetului englezesc (litera este formată din elementele a căror valoare este 1). Va trebui să determinați care este litera reprezentată de această matrice.

În acest scop veți avea la dispoziție un număr de N matrice pătratice cu 16 linii și 16 coloane care conțin reprezentări standard ale literelor. Litera corespunzătoare matricei date este "*cea mai apropiată matrice dintre cele N*".

Va trebui să definim distanța dintre două matrice. Pentru a determina o astfel de distanță, fiecărui element cu valoarea 1 din prima matrice va trebui să îi corespundă un element cu valoarea 1 din cea de-a doua matrice și fiecărui element cu valoarea 1 din cea de-a doua matrice să îi corespundă un punct din prima matrice. Așadar se va realiza o corespondență între elementele celor două matrice.

Presupunem, fără a restrânge generalitatea, că prima matrice conține mai multe elemente cu valoarea 1 decât a doua. Numărul elementelor cu valoarea 1 din prima matrice va fi notat cu p, iar numărul elementelor cu valoarea 1 din cea de-a doua va fi notat cu q. În aceste condiții, fiecărui element cu valoarea 1 din cea de-a doua matrice îi vor corespunde $[p / q]$ elemente din prima matrice. Dacă q nu este divizor al lui p, vor rămâne elemente în prima matrice care nu au corespondent. Acestora le vor corespunde elemente distincte din cea de-a doua matrice.

Cu alte cuvinte, corespondențele vor fi "*echilibrate*", în sensul că, dacă unui element dintr-o matrice îi corespund k elemente (k fiind minimul numărului de elemente corespunzătoare elementelor din matrice), atunci tuturor celorlalte elemente le vor corespunde fie k, fie $k + 1$ elemente.

Distanța dintre două matrice va fi dată de suma distanțelor dintre elementele puse în corespondență. Dacă unui element de pe poziția (i_1, j_1) în prima matrice îi corespunde un element (i_2, j_2) în cea de-a doua, atunci distanța dintre cele două elemente va fi $|i_1 - i_2| + |j_1 - j_2|$.

Evident, corespondențele trebuie alese în așa fel încât distanța dintre cele două matrice să fie cât mai mică posibil.

Date de intrare
Fișierul de intrare **OCR.IN** conține pe primele 16 linii elementele matricei care reprezintă litera care trebuie recunoscută. Pe următoarea linie se va afla numărul N al literelor date. Va urma descrierea literelor. Pentru fiecare literă vom avea 17 linii. Prima dintre ele va identifica litera (un caracter). Următoarele 16 linii vor conține descrierea literei. Valorile de pe o linie nu vor fi separate prin spații.

Date de ieșire
Fișierul de ieșire **OCR.OUT** trebuie să conțină pe o singură linie un singur caracter care reprezintă litera recunoscută.

Restricții și precizări
- $1 \le N \le 26$;
- literele descrise nu vor fi neapărat distincte;

- fiecare matrice va conține cel puțin unul și cel mult 100 de elemente cu valoarea 1;
- va exista întotdeauna o singură soluție.

Exemplu

OCR.IN

```
0000000000000000
0000000000000000
0000000000000000
0000000100000000
0000001010000000
0000010001000000
0000100000100000
0001000000010000
0001000000010000
0001111111110000
0001000000010000
0001000000010000
0001000000010000
0000000000000000
0000000000000000
0000000000000000
2
A
0000000000000000
0000000000000000
0000000100000000
0000001110000000
0000010001000000
0000100000100000
0001000000010000
0001100000110000
0001111111110000
0001100000110000
0001000000010000
0001000000010000
0001000000010000
0000000000000000
0000000000000000
0000000000000000
C
0000000000000000
0000000000000000
0000000000000000
0000111111111100
0001111000000010
0001110000000100
0001100000000000
0001100000000000
0001100000000000
0001100000000000
0001100000000000
0001100000000000
0001110000000100
0001111000000010
0000111111111100
0000000000000000
```

OCR.OUT

```
A
```

Timp maxim de execuție/test: 10 secunde

P010422: Ciobănașul

Ciobănașul *Ion* s-a decis să construiască un țarc nou pentru oile sale.

Terenul pe care ciobănașul *Ion* vrea să construiască țarcul este de formă dreptunghiulară, parcelat în $n \cdot m$ parcele și este puțin denivelat. Despre fiecare parcelă știm dacă se află la nivelul normal sau este denivelată. Țarcul trebuie să aibă o forma dreptunghiulară, laturile paralele cu laturile terenului și este susținut de patru țăruși situați în cele patru colțuri. Cei patru țăruși pot fi înfipți în pământ numai în parcele nivelate.

Pentru ciobănaș ar fi foarte important să știe câte amplasări posibile există pentru țarc, dar el este sătul de numărat oi, așa că vă cere ajutorul vostru pentru a număra dreptunghiuri.

Date de intrare

Fișierul de intrare **FOLD.IN** conține pe prima linie două numere întregi n și m separate printr-un singur spațiu care reprezintă dimensiunile terenului.

Fiecare dintre următoarele n linii conțin câte m numere, separate prin spațiu, care pot avea valorile 0 pentru teren denivelat sau 1 pentru nivel normal.

Date de ieșire

Fișierul de ieșire **FOLD.OUT** trebuie să conțină o singură linie pe care se va afla un singur număr reprezentând numărul de posibilități de amplasare.

Restricții

- $1 \le n \le 250$;
- $1 \le m \le 2000$.

Exemplu

FOLD.IN

```
3 3
0 1 1
1 1 1
1 1 1
```

FOLD.OUT

```
5
```

Timp maxim de execuție/test: 0,5 secunde

P010423: Joc

Andrei a învățat un joc matematic interesant pe care vrea să îl arate lui *Mihai*. Jocul începe cu n grămezi de pietre. Fiecare jucător aflat la mutare poate lua pietre dintr-o singură grămadă. Numărul pietrelor luate poate fi 1 sau un număr prim. Jucătorii mută alternativ, iar câștigătorul este cel care ia ultima piatră. *Andrei* se oferă să îl lase pe *Mihai* să facă prima mutare, dar *Mihai* știe că *Andrei* a învățat deja strategia jocului, așa că dorește să afle dacă are sau nu

o strategie sigură de câștig. Băieții joacă 10 jocuri consecutiv. Sarcina voastră este să îl ajutați pe *Mihai* cu răspunsul exact pentru fiecare dintre cele 10 configurații.

Date de intrare
Fișierul de intrare **GAME.IN** conține zece linii. Primul număr de pe o astfel de linie reprezintă numărul n al grămezilor, iar în continuare se află n numere care reprezintă numărul pietrelor din fiecare grămadă. Numerele de pe o linie vor fi separate prin spații.

Date de ieșire
Fișierul de ieșire **GAME.OUT** va conține câte o linie pentru fiecare dintre configurațiile din fișierul de intrare (în aceeași ordine). Valoarea de pe o linie va fi 1 dacă *Mihai* are o strategie sigură de câștig pentru configurația corespunzătoare și 0 în caz contrar.

Restricții
- $1 \le n \le 1000$;
- numărul total al pietrelor nu va depăși 2000000000 pentru o configurație.

Exemplu

GAME.IN	GAME.OUT
2 2 1	1
3 7 7 7	1
1 1000	0
2 5 5	0
4 3 3 3 3	0
2 2 1	1
3 7 7 7	1
1 1000	0
2 5 5	0
4 3 3 3 3	0

Explicație
Pentru prima configurație *Mihai* va lua o piatră din prima grămadă. Apoi, *Andrei* va lua o piatră din una dintre cele două grămezi, iar *Mihai* va lua piatra din cealaltă grămadă și va câștiga.

Pentru cea de-a doua configurație *Mihai* va lua toate cele șapte pietre din una dintre grămezi. Apoi, *Andrei* va lua un număr de pietre dintr-una dintre grămezile rămase, iar *Mihai* va lua același număr de pietre din cealaltă grămadă rămasă. Procesul va continua până în momentul în care nu va mai rămâne nici o piatră. Ultima mutare va fi efectuată cu siguranță de către *Mihai*.

Timp maxim de execuție/test: 0,5 secunde

P010424: Cifre
Se consideră un număr natural n și k cifre distincte între care se află, cu siguranță, și cifra 0. Să se determine un multiplu al lui n care conține numai cifrele date, fiecare dintre ele fiind prezentă cel puțin o dată în acest multiplu.

Date de intrare
Prima linie a fișierului de intrare **DIGITS.IN** conține valoarea n. Cea de-a doua linie a fișierului conține cifrele, neseparate prin spații.

Date de ieșire
Fișierul de ieșire **DIGITS.OUT** va conține o singură linie pe care se va afla multiplul determinat.

Restricție
- $1 \le n \le 1000$.

Exemplu

DIGITS.IN	DIGITS.OUT
11	1210
012	

Timp maxim de execuție/test: 1 secundă

P010425: Stația spațială
La stația spațială 92 a ajuns flota *Federației Planetelor Unite* care a participat la bătălia cu navale *Imperiului Cardasian*. Prima navă ajunsă este *USS Enterprise*. Celelalte nave se află pe aceeași linie, la diferite distanțe unele de altele. Pentru fiecare navă se cunoaște distanța față de nava anterioară. Din nefericire, pentru ca o navă să poată fi reparată, distanța față de navele de lângă ea trebuie să fie cel puțin egală cu o valoare x.

Așadar, anumite nave trebuie să părăsească zona. Va trebui să determinați numărul navelor care trebuie să plece, astfel încât să poată fi reparate cât mai multe nave, dar este obligatoriu ca nava *USS Enterprise* să fie reparată.

Date de intrare
Prima linie a fișierului de intrare **STATION.IN** conține numărul n al navelor, precum și valoarea x a distanței minime dintre două nave care pot fi reparate. Cea de-a doua linie a fișierului conține n - 1 numere, separate prin spații, reprezentând distanțele dintre nave.

Date de ieșire
Fișierul de ieșire **STATION.OUT** va conține pe o singură linie numărul navelor care trebuie să părăsească zona.

Restricții și precizări
- $2 \le n \le 1000$;
- $1 \le x \le 1000$;
- distanțele dintre nave sunt numere întregi cuprinse între 1 și 1000.

Exemplu

STATION.IN	STATION.OUT
5 3	2
2 3 3 2	

Timp maxim de execuție/test: 1 secundă

ALGORITMI - CALCULABILITATE

Horia Georgescu

În cadrul acestui articol, ultimul de altfel din această serie, vom aborda problema decidabilității funcțiilor folosind alte metode de calculabilitate și vom prezenta o serie de probleme nedecidabile fără însă a insista asupra demonstrațiilor acestora.

Alte modele de calculabilitate

Pentru a scurta drumul către rezultatele importante ale *teoriei calculabilității*, vom omite în continuare demonstrațiile. Cei interesați într-o expunere completă se pot adresa autorului sau pot consulta bibliografia.

Calcule pe cuvinte

Fie $A = \{s_1, s_2, ..., s_n\}$ un alfabet cu n litere. Introducem următoarea funcție $\phi_n: A^* \to N$ care stabilește pentru fiecare cuvânt $w \in A$ codificarea sa $\phi_n(w)$ ca număr natural, astfel:

$$\phi_n(w) = \begin{cases} 0, & w = \lambda \\ i_k \cdot n^k + \text{K} + i_1 \cdot n + i_0, & w = s_{i_k} s_{i_{k-1}} \text{K } s_{i_1} s_{i_0} \end{cases}$$

Exemplul 23

Fie $A = \{a, b, c\}$ cu literele precizate în această ordine. Atunci vom avea:

$\phi_3(baacb) = 2 \cdot 3^4 + 1 \cdot 3^3 + 1 \cdot 3^2 + 3 \cdot 3 + 2 = 209.$

Observăm că este vorba de ceva asemănător cu scrierea în baza n, cu deosebirea că în loc de cifrele 0, 1, ..., n - 1 se folosesc "cifrele" 1, 2, ..., n.

Motivațiile sunt următoarele:

- se evită lipsa unicității reprezentării uzuale în baza n; astfel 23 = 023 = 0023 = ... (în noua scriere nu mai apare cifra 0);
- se poate lucra și în baza 1. Astfel, dacă $A = \{a\}$, atunci: $\phi_1(a^n) = 1 \cdot 1 + ... + 1 \cdot 1 + 1 = n$.

Se impun următoarele observații:

- funcția ϕ_n depinde în mod esențial de n. Astfel, pentru exemplul de mai sus, dacă considerăm $w = baacb \in \{a, b, c, d\}$ cuvântul atașat va fi: $\phi_4(w) = 2 \cdot 4^4 + 1 \cdot 4^3 + 1 \cdot 4^2 + 3 \cdot 4 + 2 \neq \phi_3(w)$;
- funcția ϕ_n depinde și de ordinea în care apar literele în A. Nu am precizat explicit acest lucru pentru a nu încărca notația;
- aplicația ϕ_n este bijectivă.

Exemplul 24

Fie $x = 209$. Căutăm $w \in \{a, b, c\}$ cu $w = \varphi_3^{-1}(x)$.

$209 = 3 \cdot 69 + 2$
$69 = 3 \cdot 22 + 3$!!!
$22 = 3 \cdot 7 + 1$
$7 = 3 \cdot 2 + 1$
$2 = 3 \cdot 0 + 2$

Deci $209 \to (2, 1, 1, 3, 2) \to baacb$.

Prezentăm în continuare, fără demonstrație, următoarea propoziție:

Propoziția 8

Trecerea de la orice $x \in N$ la $w = \varphi_n^{-1}(x)$ se face în mod calculabil.

Fie $f: N^m \to N$. Atunci pentru orice alfabet A cu n litere într-o ordine dată, putem considera funcția $f_n = \phi_n^{-1} \circ f \circ \phi_n^m$, adică $f_n(w_1, ..., w_m) = \phi_n^{-1}(f(\phi_n(w_1), ..., \phi_n(w_m)))$.

Fie A un alfabet cu n litere într-o ordine dată și fie $f_n: (A^*)^m \to A^*$. Atunci este bine determinată funcția $f = \phi_n \circ f_n \circ (\phi_n^{-1})^m$ dată de $f(w_1, ..., w_m) = \phi_n(f_n(\phi_n^{-1}(w_1), ..., \phi_n^{-1}(w_m)))$.

Se observă în cele de mai sus că, pentru un alfabet dat cu n litere precizate într-o anumită ordine, funcțiile f_n și f sunt bine determinate una de către cealaltă, aceste determinări fiind realizate prin funcții calculabile.

Rezultă că o funcție $f: (A^*)^m \to A^*$ poate fi privită ca o funcție $f: N^m \to N$. Aceasta ne permite să vorbim de *funcții (parțial) calculabile pe cuvinte*. Mai precis, $f_n: (A^*)^m \to A^*$ se numește *(parțial) calculabilă* dacă f este (parțial) calculabilă.

Exemplul 25

Funcția $|\ |_n: N \to N$, corespunzătoare funcției: $A^* \to N$, care calculează lungimea unui cuvânt, este calculabilă:

$$|u|_n = \min_{k \le u}\left[\sum_{j=0}^{k} n^j > u\right].$$

Limbajul S_n

Limbajul S nu este adecvat pentru lucrul cu cuvinte. Astfel, ștergerea ultimei litere a unui cuvânt se face anevoios, iar instrucțiunea $v \leftarrow v + 1$ (esențială pentru lucrul cu numere) este lipsită de semnificație în lucrul pe cuvinte (de exemplu, pentru $A = \{a, b, c\}$, cuvântul *bacc* este transformat în cuvântul *bbaa*).

De aceea, pentru fiecare $n > 0$, introducem limbajul S_n, care lucrează cu variabile care sunt cuvinte într-un alfabet A cu $|A| = n$. Instrucțiunile acestui limbaj sunt următoarele:

- $v \leftarrow sv$ – se adaugă la stânga cuvântului v litera s;
- $v \leftarrow v^-$ – se șterge ultima literă a lui v; $\lambda^- = \lambda$;
- **if** v **ends** s **goto** L – dacă v se termină cu litera s, se face transfer la prima instrucțiune din program cu eticheta L; în caz contrar se trece la instrucțiuniea următoare.

Evident în instrucțiunile de mai sus trebuie să avem $v \in A^*$, $s \in A$. Prin analogie cu limbajul S, variabilele diferite de variabilele de intrare se consideră inițializate cu λ (corespunzător lui 0), se folosesc macroinstrucțiuni etc. Diferența față de S constă numai în aceea că variabilele sunt gândite ca având valori în A^*, iar forma instrucțiunilor diferă, fiind acum adecvată lucrului cu cuvinte.

Spunem că o funcție $f: (A^*)^m \to A^*$ este *(parțial) calculabilă* în S_n dacă este calculată de un program în limbajul S_n.

Deși instrucțiunile din S_n se referă la cuvinte, putem să le gândim ca referindu-se la numerele "în baza n" pe care le reprezintă acele cuvinte; de exemplu, efectul numeric al instrucțiunii $v \leftarrow sv$ este $v \leftarrow i \cdot n^{|v|_n} + v$.

Instrucțiunile din S_n sunt naturale pentru cuvinte și nenaturale pentru numere, așa cum instrucțiunile din S sunt naturale pentru numere, dar nu și pentru cuvinte.

Prezentăm câteva macroinstrucțiuni în limbajul S_n, împreună cu dezvoltările corespunzătoare (se consideră $A = \{s_1, ..., s_n\}$):

- **if** $v \neq \lambda$ **goto** L — **if** v **ends** s_i **goto** L $(1 \le i \le n)$
- $v \leftarrow \lambda$

```
[A] v ← v⁻
    if v ≠ λ goto A
```

- **goto** L

```
z ← s_1 z
if z ≠ λ goto L
```

- $v' \leftarrow v$

```
    z ← λ
    v' ← λ
[A] if v ends s_i goto B_i (1 ≤ i ≤ n)
    goto C

[B_i] (1 ≤ i ≤ n)
    v ← v⁻
    v' ← s_i v'
    z' ← s_i z'
    goto A

[C] if v ends s_i goto D_i (1 ≤ i ≤ n)
    goto E

[D_i] (1 ≤ i ≤ n)
    z ← z⁻
    v ← s_i v
    goto C
```

(se copiază mai întâi v în v' și z, iar apoi se copiază z în v).

Prezentăm în continuare, fără demonstrație, următoarea teoremă:

Teorema 2

Fie $f: N^m \to N$ (parțial) calculabilă $\Leftrightarrow \forall n$ și $A = \{s_1, ..., s_n\}$, funcția $f_n: (A^*)^m \to A^*$ este (parțial) calculabilă în S_n.

Programe Post-Turing

Fie $A = \{s_1, ..., s_n\}$ un alfabet format din n litere. Pe baza lui, vom introduce un nou limbaj: *limbajul Post-Turing* T_n.

Se folosește o bandă infinită, atât la dreapta cât și la stânga, formată din celule care pot conține fie o literă din A, fie un blanc (notat prin b sau s_0). Mai există un cap de citire / scriere care poate explora, la fiecare moment de timp, o singură celulă. Întotdeauna banda va conține numai un număr finit de celule în care apar litere din A.

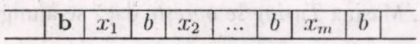
Instrucțiunile limbajului *Post-Turing* T_n sunt următoarele:

- **print** s – în celula din dreptul capului de citire / scriere se scrie litera s;
- **if** s **goto** L – dacă celula din dreptul capului de citire / scriere conține litera s, atunci se face transfer la prima instrucțiune din program cu eticheta L; în caz contrar, se trece la instrucțiunea următoare;
- **right** – capul de citire / scriere este deplasat cu o poziție la dreapta;
- **left** – capul de citire / scriere este deplasat cu o poziție la stânga.

Prin *configurația* benzii de intrare la un moment de timp oarecare înțelegem conținutul ei, împreună cu poziția capului de citire / scriere; dacă va apărea numai o porțiune din bandă, se va considera că în celulele din dreapta și stânga sa apar numai celule goale (blancuri). Pentru a indica po-

ziția capului de citire/scriere, vom figura îngroșat (bold) simbolul din celula în dreptul căreia se află.

Configurația inițială pentru un program în limbajul T_n care prelucrează datele de intrare $x_1, ..., x_m \in A^*$ este următoarea:



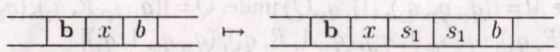
unde, așa cum am convenit, poziția capului de citire / scriere este în stânga celulei care conține valoarea x_1.

Dacă, de exemplu, $x_2 = \lambda$, atunci după x_1 vor apărea două blancuri.

Să observăm că nu putem face distincție între această configurație și cea cu $m + 1$ intrări cu $x_{m+1} = \lambda$.

Exemplul 26

Fie $A = \{s_1, s_2, s_3\}$ și $x \in A^*$. Următoarea schimbare de configurație:

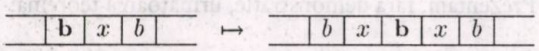


este realizată, de exemplu, de programul următor, scris în limbajul T_3:

```
[A] right
    if s_i goto A (1 ≤ i ≤ 3)
    print s_i
    right
    print s_1
[B] left
    if s_i goto B (1 ≤ i ≤ 3)
```

Exemplul 27

Fie $A = \{s_1\}$ și $x \in A^*$. Următorul program realizează schimbarea de configurație:



Vom folosi, în afară de s_1 și $b = s_0$, un simbol auxiliar $m \notin A$ (numit *marcaj*):

```
[A] right
    if b goto E
    print m
[B] right
    if s_1 goto B
[C] right
    if s_1 goto C
    print s_1
[D] left
    if s_1 goto D
    if b goto D
    print s_1
    if s_1 goto A
```

Acest program necesită următoarele explicații:

- primele trei instrucțiuni înlocuiesc primul s_1 cu m;
- următoarele patru instrucțiuni determină deplasarea la dreapta peste toți s_1, apoi peste b, apoi peste toți s_1;
- cele patru instrucțiuni care încep cu cea etichetată cu D determină deplasarea la stânga peste toți s_1 și b, până la m;
- penultima instrucțiune înlocuiește pe m cu s_1;
- ultima instrucțiune face transfer la prima, dar acum capul de citire/scriere este cu o poziție mai la dreapta.

Pentru $x = s_1s_1$, se trece succesiv prin următoarele configurații:

$$\mathbf{b}s_1s_1b \overset{1}{\mathtt{a}} bms_1b \overset{2}{\mathtt{a}} bms_1b\mathbf{b} \overset{3}{\mathtt{a}} bms_1bs_1b \overset{4}{\mathtt{a}} bms_1bs_1b \overset{5}{\mathtt{a}}$$

$$bs_1s_1bs_1b \overset{1}{\mathtt{a}} bs_1mbs_1b \overset{2}{\mathtt{a}} bs_1mbs_1\mathbf{b} \overset{3}{\mathtt{a}} bs_1mbs_1s_1b \overset{4}{\mathtt{a}}$$

$$bs_1mbs_1s_1b \overset{5}{\mathtt{a}} bs_1s_1bs_1s_1b \overset{1}{\mathtt{a}} bs_1s_1bs_1s_1b$$

Exemplul de mai sus arată necesitatea (sau cel puțin utilitatea) folosirii unor simboluri auxiliare (marcaje), pe lângă simbolurile din $A \cup \{s_0\}$.

Fie funcția parțială f_n: $(A^*)^m \to A^*$.

Spunem că un program P în limbajul T_n calculează funcția f_n dacă:

- plecându-se din configurația inițială cu valorile de intrare $x_1, ..., x_m$, programul se oprește $\Leftrightarrow f_n(x_1, ..., x_m)\downarrow$;
- la oprire, prin ștergerea de pe bandă a simbolurilor care nu sunt în A (adică $s_0 = b$ sau marcaje), rămâne $f_n(x_1, ..., x_m)$.

Prezentăm în continuare câteva macroinstrucțiuni ale limbajului T_n, împreună cu dezvoltările lor.

- **goto** L — **if** s_i **goto** L $(0 \le i \le n)$

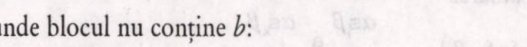
- **right to next blank**

```
[A] right
    if b goto E
    goto A
```

- **left to next blank**

```
[A] left
    if b goto E
    goto A
```

- **move block right**, care realizează schimbarea de configurație:



unde blocul nu conține b:

```
[C] left
    if s_i goto A_i (0 ≤ i ≤ n)
[A_i] (1 ≤ i ≤ n)
    right
    print s_i
    left
    goto C
[A_0] right
    print b
    left
```

- **erase a block**

```
[A] right
    if b goto E
    print b
    goto A
```

(se merge la dreapta, ștergând (înlocuind cu b) toate simbolurile până la primul b).

Prezentăm în continuare, fără demonstrație, următoarea teoremă:

Teorema 3

Pentru orice funcție parțială f_n: $(A^*)^m \to A^*$, f_n este calculată de un program în T_n dacă și numai dacă este parțial calculabilă în S_n.

Mașini Turing

La fel ca în cazul programelor *Post-Turing*, vom folosi o bandă infinită atât la dreapta, cât și la stânga, formată din celule care conțin simboluri dintr-un alfabet T, precum și simbolul b. La fiecare moment de timp pe bandă se vor afla doar un număr finit de simboluri din T. Alfabetul T conține o submulțime A. Ne va interesa în continuare cum, din cuvintele $x_1, ..., x_m \in A^*$, se poate obține un cuvânt $y \in A^*$; simbolurile din $T \setminus A$ se numesc tot *marcaje*. Capul de citire / scriere poate explora de asemenea o singură celulă. Deosebirea va consta în faptul că trecerea de la o configurație la alta nu se va mai realiza pe baza unor instrucțiuni, ci pe baza unor așa numite *stări* ale mașinii *Turing*.

O *mașină Turing* este un quadruplu $M = (K, T, q_1, Q)$ unde:

- K este alfabetul stărilor;
- T este alfabetul mașinii *Turing*;
- $q_1 \in K$ este starea inițială;
- Q este o mulțime finită de quadruple pe baza cărora se trece de la o configurație la alta.

O configurație are forma: $\overset{\alpha s \beta}{q}$, unde $\alpha, \beta \in (T \cup \{b\}^*)$, $s \in T \cup \{b\}$ este simbolul din dreptul capului de citire / scriere, iar q este starea curentă.

Prezentăm în continuare forma quadruplelor din mulțimea Q, împreună cu schimbările de configurație corespunzătoare:

- (q_i, s_j, s_k, q_l) $\quad \overset{\alpha s_j \beta}{q_i} \; a \; \overset{\alpha s_k \beta}{q_l}$
- (q_i, s_j, R, q_l) $\quad \overset{\alpha s_j s_k \beta}{q_i} \; a \; \overset{\alpha s_j s_k \beta}{q_l}$
- (q_i, s_j, L, q_l) $\quad \overset{\alpha s_k s_j \beta}{q_i} \; a \; \overset{\alpha s_k s_j \beta}{q_l}$

unde q_i este starea din care se pleacă, s_j este simbolul explorat de capul de citire / scriere, iar q_l este starea în care se ajunge; cele trei forme de quadruple determină înlocuirea simbolului s_j explorat prin s_k, deplasarea la dreapta și deplasarea la stânga a capului de citire / scriere. Vom nota în continuare $s_0 = b$.

Vom considera numai *mașini Turing deterministe*, adică în care nu există două quadruple care încep cu aceeași pereche (q_i, s_j).

Configurația inițială a benzii este următoarea:

$\overset{}{q_1} bx_1bx_2b\text{K}\; bx_m$, unde $x_1, ..., x_m \in A^*$.

Schimbările de configurație se fac conform quadruplelor din K. Mașina *Turing se oprește* dacă se ajunge într-o configurație de forma $\overset{\alpha s \beta}{q}$ și nici un quadruplu din Q nu începe cu (q_j, s_j).

Spunem că M *calculează* funcția f parțială de aritate m pe A^*, dacă plecând de la configurația inițială, M se oprește pentru $(x_1, ..., x_m) \in (A^*)^m$ dacă și numai dacă $f(x_1, ..., x_m)$ este definită; în plus, la oprire pe bandă se obține, ștergându-se toate simbolurile care nu sunt în A, tocmai $f(x_1, ..., x_m)$.

Exemplul 28

Fie $M = ((q_1, q_2, q_3), \{1\}, q_1, Q)$ unde: $Q = \{(q_1, s_0, R, q_2), (q_2, 1, R, q_2), (q_2, s_0, 1, q_3), (q_3, 1, R, q_3), (q_3, q_0, 1, q_1)\}$.

Cum $s_0 = b$, avem de exemplu:

$$\underset{q_1}{\text{b111}} \; a \; \underset{q_2}{\text{111}} \; a \; \underset{q_2}{\text{111}} \; a \; \underset{q_2}{\text{111}} \; a \; \underset{q_2}{\text{111b}} \; a \; \underset{q_3}{\text{1111}} \; a \; \underset{q_3}{\text{1111b}} \; a \; \underset{q_1}{\text{11111}}$$

(se observă că putem spune că această mașină *Turing* calculează funcția dată de $f(x) = x + 2$).

O altă reprezentare pentru o mașină *Turing* este cea sub formă de tabel, pe care ne mărginim a o prezenta pentru exemplul considerat.

	q_1	q_2	q_3
b	R, q_2	$1, q_3$	$1, q_1$
1		R, q_2	R, q_3

Prezentăm, fără demonstrație, următoarea teoremă:

Teorema 4

Orice funcție parțială este calculată de un program *Post-Turing* dacă și numai dacă este calculată și de o mașină *Turing* cu același alfabet.

Concluzie

Am introdus în acest paragraf mai multe modele de calculabilitate (de definire a noțiunii de algoritm) care au folosit fie limbaje de programare (pe numere naturale sau pe cuvinte), fie mașini matematice. Teoremele enunțate au scos în evidență echivalența lor, ceea ce confirmă *teza lui Church*.

Alte probleme nedecidabile

Includem în această secțiune alte probleme nedecidabile, neînsoțite însă de demonstrațiile corespunzătoare.

Teorema 5

Nu există un algoritm care să determine pentru orice program P din limbajul S dacă $\psi_p^{(1)}$ este sau nu total calculabilă.

Teorema spune că problema "*să se determine pentru un program oarecare dacă el se termină pentru orice valoare de intrare*" este nedecidabilă.

Observație

Rezultatul este diferit de cel din *Teorema* 1 (predicatul *HALT* nu este calculabil) deoarece aici problema are ca date de intrare programe, pe când în *Teorema* 1 problema avea ca date de intrare programe însoțite de valori de intrare.

Teorema 6

Fie f o funcție (parțială) calculabilă de o variabilă. Atunci nu există un algoritm pentru următoarea problemă: *să se determine pentru orice program P dacă* $\psi_P^{(1)} = f$.

Facem următoarea observație legată de această teoremă: să considerăm o funcție calculabilă simplă, de exemplu, cea dată de $f(x) = x + 1$. Presupunem că cerem mai multor persoane să scrie programe în limbajul S care să calculeze această funcție. Teorema de mai sus ne spune că *nu există un algoritm care să verifice că aceste programe produc rezultatul dorit. Deci, cel puțin din punct de vedere al corectării programelor, profesorul nu poate fi înlocuit de calculator.*

Teorema 7

Nu există un algoritm pentru următoarea problemă: fiind date $u, v \in N$, să se determine dacă $\Phi_u^{(1)} = \Phi_v^{(1)}$.

Teorema de mai sus arată *că problema echivalenței a două programe este nedecidabilă.*

Încheiem lista problemelor nedecidabile cu *problema de corespondență a lui Post*, formulată în anul 1946, problemă cu aplicații în teoria limbajelor formale.

Un sistem Post peste un alfabet A constă dintr-un număr finit k de tipuri de dominouri de forma:

$$\begin{array}{|c|}\hline u_i \\ \hline v_i \\ \hline\end{array},$$

cu $u_i, v_i \in A^*$, $i = 1, ..., k$ (săgeata de pe dominiouri arată că nu le putem răsturna; nu o vom mai figura în continuare).

Există o infinitate de dominouri de fiecare tip.

Un sistem Post are soluție dacă există o înșiruire de dominouri, astfel încât concatenarea cuvintelor aflate în partea superioară și concatenarea cuvintelor aflate în partea inferioară să producă același cuvânt.

Exemplul 29

Fie $A = \{a, b\}$. Atunci sistemul *Post* următor:

$$\left\{ \begin{array}{|c|}\hline a \\ \hline aa \\ \hline\end{array}, \begin{array}{|c|}\hline bb \\ \hline b \\ \hline\end{array}, \begin{array}{|c|}\hline a \\ \hline bb \\ \hline\end{array} \right\},$$

are soluția a^2b^4 dată de:

$$\begin{array}{|c|c|c|c|}\hline a & a & bb & bb \\ \hline aa & bb & b & b \\ \hline\end{array}.$$

Problema de corespondență a lui *Post* are următorul enunț: există un algoritm care să determine pentru orice sistem Post dacă el are sau nu soluții?

Teorema de mai jos arată că *problema de corespondență a lui Post nu este decidabilă.*

Teorema 8

Fie A un alfabet cu $|A| \geq 2$. Atunci nu există un algoritm care pentru orice sistem *Post P* peste A să determine dacă P are soluție.

Scurt istoric al teoriei calculabilității

Noțiunea de algoritm în accepțiunea uzuală, cunoscută de sute de ani, a fost și este suficientă pentru a elabora algoritmi pentru diferite probleme, inclusiv de a-i implementa în diferite limbaje de programare.

Totuși, în anii 1930, dezvoltarea logicii matematice, precum și perspectiva imediată a apariției calculatoarelor, a făcut ca matematicienii să încerce să găsească o definiție riguroasă a conceptului de algoritm. Necesitatea acestui lucru a apărut deci prin prisma evoluției matematicii, dar și pentru a răspunde la o întrebare naturală: "e foarte bine că vor fi construite calculatoare, dar care sunt limitele prelucrărilor pe care le pot realiza?".

Charles Babbage, plecând de la o idee a lui *Leibniz*, a încercat să construiască un raționament matematic care să folosească în mod mecanic simbolurile algebrice. Această încercare a fost concretizată în mașina sa în greutate de 3 tone, bazată pe metoda diferențelor finite.

Punctul de plecare al cercetărilor ce au condus la definirea conceptului de algoritm este considerat programul matematic inițiat de *David Hilbert*. *Hilbert* a propus o metodă axiomatică de lucru cu numerele naturale, în care acestea sunt gândite ca obiecte între care există relați descrise printr-un sistem necontradictoriu de axiome. Axiomele și teoremele sunt reprezentate prin secvențe finite de simboluri, iar demonstrațiile trebuiesc făcute conform unor "reguli mecanice". În 1922 a propus celebra *Entscheidungsproblem* (problema deciziei) care constă în a găsi o modalitate de a testa într-un număr finit de pași dacă o expresie formală poate fi dedusă dintr-un sistem dat de axiome.

Zece ani mai târziu, s-a arătat însă că problema deciziei nu poate avea un răspuns afirmativ; era necesar să se precizeze ce se înțelege prin funcție calculabilă și prin procedură de decizie.

În anul 1931, *Kurt Gödel* a dovedit incompletitudinea aritmeticii: "există propoziții din aritmetică pentru care nu se poate demonstra dacă sunt adevărate sau false". Pentru aceasta, el a introdus noțiunile de funcție recursivă și funcție calculabilă.

Denumirile au variat de-a lungul anilor, de aceea prezentăm în continuare accepțiunile actuale.

Vom numi funcții inițiale următoarele trei funcții, evident calculabile:

- *funcția succesor* $s: N \to N$ dată de $s(x) = x + 1$;
- *funcția nulă* $n: N \to N$ dată de $n(x) = 0$;
- *funcțiile proiecții* $u_i^n: N^n \to N$ $(i = 1, 2, ..., n)$ date de $u_i^n(x_1, x_2, ..., x_n) = x_i$.

Definiție

O funcție se numește *primitiv recursivă* dacă se obține din funcțiile inițiale aplicând de un număr finit de ori operația de compunere și recursia (1).

Evident, orice funcție primitiv recursivă este calculabilă.

Teorema 9

Există funcții calculabile, care nu sunt primitiv recursive.

Lucrările lui *Gödel* au stat la baza cercetărilor întreprinse de *Alonzo Church*, *Stephen Kleene*, *Alan Turing* și *Emil Post*. Fiecare dintre ei a definit un concept de calculabilitate, adică un concept de algoritm. Pe baza acestui concept au abordat problema de decizie a lui *Hilbert*, demonstrând că există probleme nedecidabile. Unele dintre aceste abordări au fost prezentate în acest articol, într-o formă posibilă după decenii de simplificări și rafinări.

Echivalența diferitelor concepte de algoritm introduse a fost esențială în acceptarea *tezei lui Church*, care ține de domeniul *metamatematicii*.

Note biografice

Charles Babbage (*Teignmouth* 1792 - *Londra* 1871)
Absolvent al Universității Cambridge. A avut preocupări de calcul diferențial și integral, dar în special de logică matematică. A conceput două mașini de calcul, una bazată pe diferențe finite, iar cealaltă analitică, aceasta din urmă fiind unanim considerată ca precursoarea calculatoarelor moderne. Membru al *Societății Regale*.

Augusta Ada King, contesă de Lovelace (1815 - 1852)
Fiica poetului *Lord George Gordon Byron*, colaboratoare a lui *Charles Babbage*. Este considerată prima femeie informatician din lume. A dat numele limbajului de programare *Ada*, care a introdus concepte noi în teoria programării.

David Hilbert (*Königsberg* 1862 - *Götingen* 1943)
Considerat de mulți ca cel mai mare matematician al epocii moderne, a avut contribuții importante în domeniile teoriei algebrice a numerelor, fundamentelor geometriei, analizei, fizicii teoretice și fundamentelor matematicii. Cele 23 de probleme propuse la *Congresul matematicienilor de la Paris* (1900), dintre care ultima a primit răspuns doar în 1970, continuă să stimuleze studiile teoretice.

Kurt Gödel (*Brno* 1906 - *Princeton* 1978)
Lucrările sale cuprind arii vaste: fizică, logică și filozofia matematicii. Celebru în special pentru teorema de incompletitudine (1933) care îi poartă numele. Membru al *Cercului de Filozofie* de la *Viena*, fuge la *Princeton* după instalarea regimului nazist.

Alonzo Church (*Washington* 1903 - *Götingen* 1995)
A urmat studiile la *Princeton*, apoi a profesat la *UCLA*. Cunoscut ca matematician, logician și filozof. Cele mai importante rezultate obținute sunt teorema care îi poartă numele (care afirmă că nu există o procedură de decizie pentru aritmetică) și *teza lui Church*, prezentată în acest articol.

Stephen Cole Kleene (*Hartford* 1909 - *Madison* 1994)
Matematician ilustru, cu contribuții majore în domeniile logicii matematice, calculabilității și limbajelor formale. Contribuțiile sale în domeniul funcțiilor recursive au avut consecințe notabile, dintre care menționăm demonstrația lui *Matijasevics* (1970) a nedecidabilității celebrei probleme a zecea a lui *Hilbert* privind existența unei soluții întregi pentru polinoamele cu coeficienți întregi.

Emil Leon Post (*Augustow* 1897 - *New York* 1954)
Își obține doctoratul la *Universitatea din Columbia* și apoi predă la *Universitatea Cornell* și la alte universități din *New York*. Membru în *American Mathematical Society* și fondator al *Association for Symbolic Logic*. Este considerat ca precursor al teoriei moderne a demonstrației.

Alan Turing (*Londra* 1912 - *Wilmslow* 1954)
A absolvit *King's College*, *Cambridge*, apoi s-a mutat la *Princeton* pentru a lucra cu *Church*. Aici a publicat celebrul său articol privind problema de decizie a lui *Hilbert* în care a introdus modelul de mașină *Turing*, model de referință și astăzi. Întors în *Anglia*, a lucrat ca specialist în descifrarea codurilor la *Foreign Oce* în timpul celui de al doilea război mondial. După război a participat la producerea unei mașini automate de calcul.

A formulat întrebarea "Presupunând că un calculator poate să gândească, cum putem să ne dăm seama (If a computer could think, how would we tell)?". Întrebarea s-a concretizat în *testul lui Turing*: "se pun întrebări unei persoane și unui calculator, la care trebuie să răspundă printr-un text. Poate un arbitru să nu fie capabil să identifice sursa răspunsurilor?". Deocamdată nici un calculator nu a trecut *testul lui Turing* (premiul de 100.000 lire sterline își așteaptă în continuare câștigătorul...). Mai menționăm că *Turing* este considerat părintele inteligenței artificiale.

Bibliografie

1. Martin D. Davis, Elaine J. Weyuker, *Computability, Complexity and Languages*, Academic Press, 1983
2. Christos H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994
3. Jean Luc Chambert, *A History of Algorithms*, Springer Verlag, 1999
4. http://turnbull.mcs.st-and.ac.uk/history/Mathematicians

Dl. prof. dr. Horia Georgescu este cadru didactic la Universitatea București, fondator și director științific al GInfo și poate fi contactat prin e-mail la adresa g_horia@hotmail.com.

$$x_n = f(n, x_{n-1}, x_{n-2}, ..., x_0)$$

Șiruri RECURENTE

Eugen Nodea

Deoarece este imposibil să evităm folosirea șirurilor recurente în informatică, în cadrul acestui articol vom trata problema șirurilor recurente și ne vom concentra atenția asupra rezolvării recurențelor liniare de ordinul 1 și 2, acestea fiind întâlnite foarte des și vom da exemple de șiruri recurente celebre împreună cu proprietățile lor.

Fie o funcție $f: N^* \to M$ ce definește un șir $(x_n)_{n \in N}$, cu $M \subseteq R$.

Studiul șirurilor rezidă în evidențierea unor caracteristici specifice: *mărginirea*, *monotonia*, *convergența*, *calculabilitatea*, *periodicitatea*.

În acest articol ne vom referi la calculabilitatea șirurilor, o modalitate interesantă de definire a șirurilor reprezentând-o descrierea recurentă a acestora pe baza unor formule recurente.

Vom porni de la definiția: *Numim formulă de recurență o formulă care exprimă orice termen al șirului, de la un rang oarecare, pe baza unuia sau mai mulți precedenți.*

O relație de recurență poate fi dată ca o egalitate în formă implicită, explicită sau o inegalitate.

Forma generală *implicită* a unei relații de recurență este:

$F(n, x_n, x_{n-1}, ..., x_0) = 0, \forall n \in N$, unde $F: N \times M^{n+1} \to M$.

Forma generală *explicită* a unei relații de recurență este:

$x_n = f(n, x_{n-1}, x_{n-2}, ..., x_0) = 0, \forall n \in N$, unde $f: N \times M^n \to M$.

Un caz extrem de important îl reprezintă recurențele de ordinul k, $k \geq 1$, $k \in N$, recurențele de ordinul k fiind scrise în cele două forme, implicită și explicită, astfel:

- $F(n, x_{n+k}, x_{n+k-1}, ..., x_0) = 0, \forall n \in N, n \geq k$;
- $x_{n+k} = f(n, x_{n+k-1}, x_{n+k-2}, ..., x_0), \forall n \in N, n \geq k$.

În continuare prezentăm câteva exemple de relații în formă implicită:

- $F(n, x_{n+1}, x_n) = 0$ – relație de recurență de ordinul 1;
- $F(n, x_{n+2}, x_{n+1}, x_n) = 0$ – relație de recurență de ordinul 2.

Aceste relații exprimate în formă explicită, sunt:

- $x_{n+1} = f(n, x_n)$ – relație de recurență de ordinul 1;
- $x_{n+2} = f(n, x_{n+1}, x_n) = 0$ – relație de recurență de ordinul 2.

Recurențele pot fi: *liniar omogene*, *liniar neomogene* sau *omografice*.

Exemplificăm relațiile recurente de ordinul 1 prin următoarele relații:

- liniar omogene: $x_{n+1} = a \cdot x_n, \forall n \in N$;
- liniar neomogene: $x_{n+1} = a \cdot x_n + b, \forall n \in N$;
- omografice: $x_{n+1} = (a \cdot x_n + b) / (c \cdot x_n + d), c \cdot x_n + d \neq 0, \forall n \in N$.

Pentru a rezolva recurențele avem la dispoziție următoarele trei metode:

- metoda *substituției*;
- metoda *iterației*;
- metoda *master*.

O amplă analiză a metodelor de rezolvare a recurențelor se găsește în cartea *Introducere în algoritmi, T. H. Cormen, C. E. Leiserson, R. L. Rivest*. În acest articol ne propunem rezolvarea punctuală a unor tipuri de recurențe.

Recurențe liniare de ordinul 1

Considerăm mulțimea șirurilor definite prin relația de recurență:

- $x_0 = c$,
- $x_{n+1} = a \cdot x_n + b, \forall n \in N$ și $a, b, c \in R$.

Pentru cazurile particulare:

- $a = 1, b \neq 0$ se obține o progresie aritmetică $x_{n+1} = x_n + b$;
- $a \neq 0, a \neq 1, b = 0$ se obține progresia geometrică cu rația a.

Se pune problema determinării termenului general al șirului $(x_n)_{n \in N}$ prin simplificarea relației de recurență.

Primul pas constă în determinarea unei valori $y \in R$ astfel încât: $x_{n+1} + y = a \cdot (x_n + b)$.

Din relația de recurență avem:

- $x_1 + y = a \cdot (x_0 + b)$ și
- $x_1 = a \cdot x_0 + b$.

Din cele două relații rezultă că $y = b / (a - 1)$.

Notăm cu t_n expresia $x_{n+1} + y$ și obținem progresia geometrică $t_n = a \cdot t_{n-1}$. Deci, $t_n = a^n \cdot t_0$ și prin înlocuire obținem $x_n = a^n \cdot (c + b / (a - 1)) - b / (a - 1)$.

Recurențe liniare omogene de ordinul 2

Considerăm mulțimea șirurilor definite prin relația de recurență:

- $x_{n+2} = a \cdot x_{n+1} + b \cdot x_n$, (*)
- $x_0 = c_0$, (**)
- $x_1 = c_1$, $\forall n \in N$, $a, b, c_0, c_1 \in R$ și $b \neq 0$.

Pentru a putea rezolva astfel de recurențe avem nevoie de câteva teoreme pe care le prezentăm în continuare fără a insista asupra demonstrațiilor.

Teoremă

Dacă șirurile $(\alpha_n)_{n\in N}$ și $(\beta_n)_{n\in N}$ îndeplinesc condiția (*), atunci șirul cu termenul general $a \cdot \alpha_n + b \cdot \beta_n$ îndeplinește aceeași condiție.

Teoremă

Dacă α este o rădăcină a ecuației $r^2 = a \cdot r + b$, atunci α^n satisface relația (*).

Teoremă

Dacă ecuația caracteristică $r^2 = a \cdot r + b$ are rădăcinile reale distincte r_1, r_2 atunci sistemul:

$$\begin{cases} a + b = c_0 \\ a \cdot r_1 + b \cdot r_2 = c_1 \end{cases},$$

admite soluție unică.

Teoremă

Dacă ecuația caracteristică $r^2 = a \cdot r + b$ admite o rădăcină dublă α, atunci șirul cu termen general $n \cdot \alpha^n$ satisface relația (*).

Teoremă

Dacă ecuația caracteristică $r^2 = a \cdot r + b$ admite o rădăcină dublă α, atunci șirul care satisface relațiile (*) și (**) are termenul general de forma $x_n = a \cdot \alpha^n + b \cdot n \cdot \alpha^n$

Exemplu

Să se determine șirul $(f_n)_{n\in N}$ definit astfel (șirul lui *Fibonacci*):

- $f_0 = 0$,
- $f_1 = 1$,
- $f_{n+2} = f_{n+1} + f_n$, $\forall n \in N$, $n \geq 2$.

Ecuația caracteristică asociată șirului este $r^2 = r + 1$ și are rădăcinile reale: $r_1 = (1-\sqrt{5})/2$ și $r_2 = (1+\sqrt{5})/2$.

Șirul are termenul general de forma:

$$f_n = a \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n + b \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n$$

Condițiile inițiale impun:

$$a = \frac{1}{\sqrt{5}}, b = -\frac{1}{\sqrt{5}},$$

de unde rezultă în final formula lui *Binet*:

$$f_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n\right].$$

Concluzie

În determinarea unui anumit termen calculabil printr-o relație de recurență, abordarea recursivă pare cea mai accesibilă. Totuși, cu minime cunoștințe matematice, putem implementa un algoritm care să nu necesite dezvoltarea termenilor precedenți.

Prezentăm în cele ce urmează șiruri recurente celebre.

Șirul lui Fibonacci

Definim șirul *Fibonacci* după următoarea relație recurentă:

- $f_0 = 0, f_1 = 1$,
- $f_n = f_{n-1} + f_{n-2}$, $\forall n \in N$, $n \geq 2$.

Astfel, se obține următorul șir de numere: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377 ...

Termenul de rang k al șirului se poate determina cu ajutorul următoarei formule:

$$f_k = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^k - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^k\right].$$

Proprietăți

- $\sum_{i=1}^{n} f_i = f_1 + f_2 + f_3 + \ldots + f_n = f_{n+2} - 1$,
- $f_1 + f_3 + f_5 + \ldots + f_{2n-1} = f_{2n}$,
- $f_2 + f_4 + f_6 + \ldots + f_{2n} = f_{2n+1} - 1$,
- $f_0^2 + f_1^2 + f_2^2 + f_3^2 + \ldots + f_n^2$,
- $f_{n+m} = f_{n-1} \cdot f_m + f_n \cdot f_{m+1}$,
- $f_{2n} = f_{n-1} \cdot f_n + f_n \cdot f_{n+1}$,
- $f_{2n} = f_{n+1}^2 - f_{n-1}^2$,
- $f_{3n} = f_{n+1}^3 + f_n^3 - f_{n-1}^3$,
- $f_n^2 = f_{n-1} \cdot f_{n+1} + (-1)^{n+1}$,
- $f_1 \cdot f_2 + f_2 \cdot f_3 + \ldots + f_{2n-1} \cdot f_{2n} = f_{2n}^2$,
- $f_1 \cdot f_2 + f_2 \cdot f_3 + \ldots + f_{2n} \cdot f_{2n+1} = f_{2n+1}^2 - 1$,
- $f_1 + 2 \cdot f_2 + 3 \cdot f_3 + \ldots + n \cdot f_n = n \cdot f_{n+2} - f_{n+3} + 2$,
- $n \cdot f_1 + (n-1) \cdot f_2 + \ldots + 2 \cdot f_{n-1} + f_n = f_{n+4} - (n+3)$.

Divizibilitate

- Termenul *Fibonacci* f_n este par dacă și numai dacă n este divizibil cu 3.
- Termenul *Fibonacci* f_n este divizibil cu 3 dacă și numai dacă n este divizibil cu 4.
- Termenul *Fibonacci* f_n este divizibil cu 4 dacă și numai dacă n este divizibil cu 6.
- Termenul *Fibonacci* f_n este divizibil cu 5 dacă și numai dacă n este divizibil cu 5.
- Termenul *Fibonacci* f_n este divizibil cu 7 dacă și numai dacă n este divizibil cu 8.

- Termenul *Fibonacci* f_n este divizibil cu 16 dacă și numai dacă n este divizibil cu 12.
- Termenul *Fibonacci* f_{2n} se divide cu f_n.
- Termenul *Fibonacci* f_n este prim numai dacă și n este prim cu excepția lui f_4 care este egal cu 3. Nu orice n prim implică faptul ca f_n să fie prim.
- Oricare al n-lea termen din șir se divide prin f_n.
- Dacă n este număr compus, atunci și f_n este tot un număr compus (un număr este compus dacă nu este prim).
- Dacă m este divizibil cu n, atunci și f_m este divizibil cu f_n.

Alte proprietăți

- Dacă notăm cu $\Phi = 1+\sqrt{5} \approx 1.$ (se poate aproxima:

$$f_n \approx \text{rotunjire}(\Phi^n / \sqrt{5}).$$

- Numărul de cifre ale lui f_n poate fi determinat astfel:

$$Nc = \log(\Phi^n / \sqrt{5}) = n \cdot \log(\Phi) - \log(5)/2.$$

- Seria formată din ultima cifră a numerelor din șirul lui *Fibonacci* se repetă după un ciclu de 60 de numere.
- Seria formată din ultimele 2 cifre ale numerelor din șirul lui *Fibonacci* se repetă după un ciclu de 300 de numere.

Triunghiul lui Pascal

Pentru următoarea dispunere a *triunghiului lui Pascal* se observă că suma elementelor de pe coloana de rang k reprezintă de fapt al k-lea termen al șirului *Fibonacci*.

0	1	2	3	4	5	6	7	8	9	10
0	1									
1		1	1							
2			1	2	1					
3				1	3	3	1			
4					1	4	6	4	1	
5						1	5	10	10	5
6							1	6	15	20
7								1	7	21
8									1	8
9										1
10										
	1	1	2	3	5	8	13	21	34	55

Pentru următoarea dispunere a *triunghiului lui Pascal* se observă că suma pe diagonala de rangul k reprezintă de fapt al k-lea termen al șirului lui *Fibonacci*. De exemplu, $S_4(8) = 1 + 6 + 10 + 4 + 0 + 0 + 0 + 0 = 21 = f_8$.

	0	1	2	3	4	5	6	7	8
0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0
2	1	2	1	0	0	0	0	0	0
3	1	3	3	1	0	0	0	0	0
4	1	4	6	4	1	0	0	0	0
5	1	5	10	10	5	1	0	0	0
6	1	6	15	20	15	6	1	0	0
7	1	7	21	35	35	21	7	1	0
8	1	8	28	56	70	56	28	8	1

Mai mult, $f_n = \sum_{k=1}^{n} C_{k-1}^{n-k}$.

Șirul lui Lucas

Definim *șirul lui Lucas* cu ajutorul următoarelor relații:

$$L_0 = 2,\ L_1 = 1,\ L_n = L_{n-1} + L_{n-2},\ \forall n \in N,\ n \geq 2.$$

Astfel, obținem șirul: 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, ...

Proprietăți

- $\sum_{i=1}^{n} L_i = L_1 + L_2 + L_3 + ... + L_n = L_{n+2} - 1,$
- $L_1 + L_3 + L_5 + ... + L_{2n-1} = L_{2n},$
- $L_2 + L_4 + L_6 + ... + L_{2n} = L_{2n+1} - 2,$
- $L_n = f_{n-1} + f_{n+1},$
- $5 \cdot f_n = L_{n-1} + L_{n+1},$
- $L_n = \Phi^n + (-\Phi)^n.$

Super șirul lui Catalan

Definim *super șirul Catalan* cu ajutorul următoarelor relații:

- $T_1 = 1,\ T_2 = 1,$
- $T_n = (3 \cdot (2 \cdot n - 3) \cdot T_{n-1} - (n - 3) \cdot T_{n-2}) / n,\ \forall n \in N,\ n \geq 3.$

Astfel, se obține următorul șir de numere: 1, 1, 3, 11, 45, 197, ...

Reamintim și șirul numerelor lui Catalan, termenul general fiind determinat cu relația:

$$C_n = \frac{1}{n+1} \cdot C_{2n}^n = \frac{1}{n+1} \cdot \frac{(2n)!}{n!^2} = \frac{(2n)!}{(n+1)!n!}.$$

Astfel, se obține următorul șir de numere: 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900,...

Proprietăți ale numerelor lui Catalan

- $C_{n+1} = \frac{2 \cdot (2 \cdot n) + 1}{n+2} \cdot C_n,$
- $C_n = \sum_{k=1}^{n-1} C_k \cdot C_{n-k},$
- Numărul de parantezări distincte $P(n) = C_{n-1}$,
- Numărul de triangularizări ale unui poligon convex cu n vârfuri, adică împărțirea unui poligon în n triunghiuri disjuncte = C_{n-2}, $n \geq 3$.
- Numerele lui Catalan în triunghiul lui Pascal,

$$C_{2n-1}^n - C_{2n-1}^{n+1} = C_{2n}^n - C_{2n}^{n+1} = \frac{1}{n+1} C_{2n}^n.$$

- Numărul de arbori binari cu n noduri este C_n.

Bibliografie

1. Dan Brânzei și colab., *Șiruri recurente în liceu*, Editura Gil, 1996
2. I. Munteanu, D. Popa, *Metoda șirurilor recurente*, Editura Gil
3. Richard Stanley, *Exercises on Catalan Number*, Cambridge University Press, 1999
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introducere în algoritmi*, Computer Libris Agora, 2000
5. **http://math.ournet.md**
6. **http://mathworld.wolfram.com**
7. **http://www.ginfo.ro**

Dl. profesor Eugen Nodea este cadru didactic la Colegiul Național "Tudor Vladimirescu" din Târgu-Jiu. Poate fi contactat prin e-mail la adresa **enodea@utgjiu.ro**

Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

În cadrul acestui articol vom continua prezentarea funcțiilor utilizate pentru prelucrarea și publicarea imaginilor dinamice și vom prezenta, de asemenea, funcțiile de acces la baze de date MySQL.

Scrierea textului pe imagini

În continuare vom prezenta câteva funcții utilizate pentru a scrie un text sau un caracter pe o imagine. Nu vom prezenta însă modul în care se poate încărca și folosi un anumit tip de caractere (*font*).

Funcția imagechar

Acestă funcție scrie pe o imagine primul caracter al unui șir de caractere.

Funcția `imagechar` are șase parametri:

- primul parametru este identificatorul de acces la imaginea pe care se va desena.
- cel de-al doilea parametru reprezintă identificatorul de acces la *font*-ul care va fi folosit pentru a scrie caracterul pe imagine. Acest identificator este de tip număr întreg și poate avea valorile predefinite 1, 2, 3, 4 sau 5 sau poate fi un număr returnat de o funcție care încarcă un font de pe disc.
- cel de-al treilea parametru reprezintă coordonata *x* a colțului stânga-sus de unde va începe scrierea caracterului;
- cel de-al patrulea parametru reprezintă coordonata *y* a colțului stânga-sus de unde va începe scrierea caracterului;
- cel de-al cincilea parametru reprezintă șirul de caractere din care va fi scris doar primul caracter;
- cel de-al șaselea parametru reprezintă culoarea cu care se va scrie pe imagine.

În continuare vă prezentăm un exemplu de utilizare a acestei funcții, iar în figura 1a) se poate observa efectul execuției *script*-ului.

```
<?PHP
  header("Content-type: image/png");
  $im = imagecreatefromjpeg("images/ginfo.jpg");
  $gray = imagecolorallocate($im, 127,127,127);
  imagechar($im, 5, 75, 55, 'a', $gray);
  imagepng($im);
  imagedestroy($im);
?>
```

Funcția imagecharup

Această funcție scrie un caracter vertical pe o imagine. Are aceeași parametri ca funcția anterioară.

În continuare vă prezentăm un exemplu de utilizare a acestei funcții, iar în figura 1b) se poate observa efectul execuției *script*-ului.

```
<?PHP
  header("Content-type: image/png");
  $im = imagecreatefromjpeg("images/ginfo.jpg");
  $gray = imagecolorallocate($im, 127,127,127);
  imagechar($im, 5, 70, 65, 'a', $gray);
  imagepng($im);
  imagedestroy($im);
?>
```

Funcția imagestring

Această funcție scrie un text pe o imagine și are aceeași parametri ca funcția `imagechar`.

În continuare vă prezentăm un exemplu de utilizare a acestei funcții, iar în figura 2a) se poate observa efectul execuției *script*-ului.

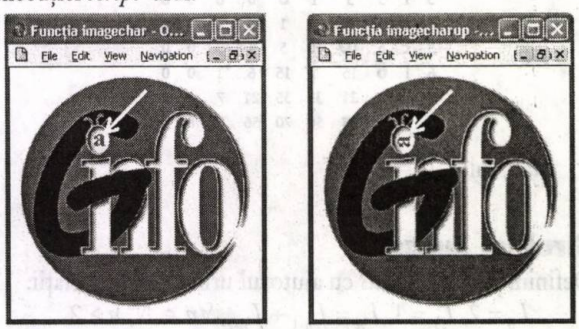


Figura 1: a) Funcția imagechar
b) Funcția imagecharup

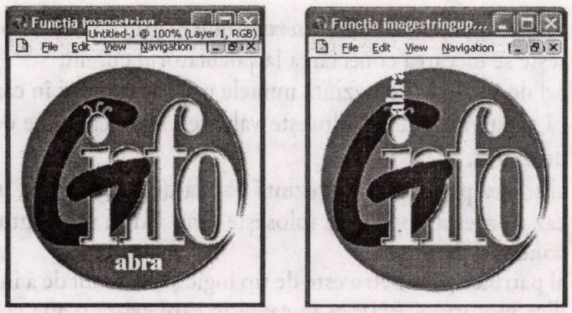


Figura 2: a) Funcția imagestring
b) Funcția imagestringup

```
<?PHP
  header("Content-type: image/png");
  $im = imagecreatefromjpeg("images/ginfo.jpg");
  $wh = imagecolorallocate($im, 255, 255, 255);
  imagestring($im, 5, 100, 190, 'abra', $wh);
  imagepng($im);
  imagedestroy($im);
?>
```

Funcția imagestringup

Această funcție scrie vertical un text pe o imagine și are aceeași parametri ca funcția `imagechar`.

În continuare vă prezentăm un exemplu de utilizare a acestei funcții, iar în figura 2b) se poate observa efectul execuției *script*-ului.

```
<?PHP
  header("Content-type: image/png");
  $im = imagecreatefromjpeg("images/ginfo.jpg");
  $wh = imagecolorallocate($im, 255, 255, 255);
  imagestringup($im, 5, 70, 50, 'abra', $wh);
  imagepng($im);
  imagedestroy($im);
?>
```

Alte prelucrări ale imaginilor

În cadrul acestui paragraf vom prezenta alte funcții utilizate pentru a prelucra în mod dinamic imagini.

Funcția imagerotate

Această funcție realizează rotirea unei imagini în sens trigonometric cu un unghi primit ca parametru. Imaginea rezultată în urma rotirii va avea dimensiunile egale cu dimensiunile celui mai mic dreptunghi ale cărui laturi sunt paralele cu axele *Ox* și *Oy* și care conține imaginea rotită.

Funcția are trei parametri:

- primul reprezintă identificatorul de acces la imaginea;
- al doilea este de tip real și reprezintă unghiul cu care se va roti imaginea;
- al treilea parametru este de tip întreg și reprezintă culoarea cu care se va umple dreptunghiul care va conține imaginea rotită.

Funcția `imagerotate` returnează un identificator de acces la imaginea rotită.

În continuare vă prezentăm un exemplu de utilizare a acestei funcții, iar în figura 3 se poate observa efectul execuției *script*-ului.

```
<?PHP
  header("Content-type: image/png");
  $im = imagecreatefromjpeg("images/ginfo.jpg");
  $gray = imagecolorallocate($im, 192, 192, 192);
  $imrot = imagerotate($im, 45, $gray);
  imagepng($imrot);
  imagedestroy($imrot);
?>
```

Funcția imagetruecolorpalette

Această funcție realizează transformarea unei imagini care nu folosește o paletă de culori într-o imagine care folosește o paletă de culori.

Funcția `imagetruecolorpalette` are trei parametri:

- primul reprezintă identificatorul de acces al imaginii care va fi transformată;
- cel de-al doilea parametru este de tip logic și reprezintă modul în care se va face transformarea: dacă imaginea va fi fără zgomot (valoarea logică **`FALSE`** implică o aproximare mai slabă a culorilor); dacă imaginea va fi cu zgomot (valoarea **`TRUE`** implică o aproximare mai bună a culorilor).

Funcția jpeg2wbmp

Această funcție transformă o imagine din formatul *JPEG* în formatul *WBMP*.

Funcția are cinci parametri. Primul parametru reprezintă calea spre fișierul de tip *JPEG* care va fi transformat. Al doilea parametru reprezintă calea spre fișierul care va conține rezultatul transformării.

Următorii doi parametri sunt de tip întreg și reprezintă înălțimea și lățimea noii imagini, iar ultimul parametru, tot de tip întreg, reprezintă pasul cu care se va parcurge imaginea pentru a fi transformată.

Funcția png2wbmp

Această funcție transformă o imagine din formatul *PNG* în formatul *WBMP*. Are aceeași parametri ca funcția anterioară, cu observația că primul parametru reprezintă calea spre un fișier de tipul *PNG*.

Figura 3: Funcția imagerotate

Publicarea imaginilor

După cum s-a putut observa, în cadrul *script*-urilor prezentate ca exemplu, am folosit funcția `imagepng`. Această funcție este folosită pentru a scrie la ieșirea standard o imagine în formatul *PNG*.

În continuare vom prezenta câteva funcții utilizate pentru a scrie la ieșirea standard o imagine în diferite formate, dar mai întâi vom prezenta funcția `imageinterlace`, o funcție foarte importantă în cazul în care doriți ca *site*-ul vostru să aibă o navigabilitate bună, chiar dacă nu s-au încărcat integral toate elementele grafice prezente în cadrul lui.

Funcția imageinterlace

Această funcție setează proprietatea *interlace* al unei imagini la valoarea logică precizată ca parametru sau la valoarea logică **TRUE** dacă acest parametru lipsește.

Funcția are doi parametri. Primul parametru reprezintă identificatorul de acces la imaginea pentru care se dorește setarea proprietății *interlace*, iar al doilea parametru este de tip logic și reprezintă valoarea pe care o va avea această proprietate. Cel de-al doilea parametru poate lipsei, dar proprietatea *interlace* a unei imagini va primi valoarea **TRUE**.

Ce înseamnă de fapt că o imagine are proprietatea interlace? Ei bine, în cazul în care aveți o viteză de acces la *Internet* mică sau imaginea pe care doriți să o încărcați este mare, nu este nevoie să așteptați încărcarea întregii imagini pentru a vedea ce este în ea. O astfel de imagine se încarcă adaptiv, și în momentul în care s-a încheiat transferul imaginii către *browser*-ul utilizatorului, pe imagine se pot observa toate detaliile. De fapt, teoretic vorbind, o imagine este împărțită în subimagini de aceeași dimensiune și calitate diferită (din ce în ce mai bună), care suprapuse dau ca rezultat imaginea inițială. Practic nu se întâmplă chiar așa, dar efectul obținut este cel precizat anterior.

Publicare în diverse formate

Pentru a publica o imagine generată dinamic se poate utiliza una dintre funcțiile: `imagegif`, `imagepng`, `imagejpeg` sau `imagewbmp`. Aceste funcții primesc ca parametru un identificator de imagine și tipăresc la ieșirea standard imaginea în format *GIF*, *PNG*, *JPEG* sau *WBMP*.

Primele trei funcții se folosesc pentru a genera o imagine care să poată fi vizualizată într-un *browser* încărcat de pe un calculator, iar ultima se folosește pentru ca imaginea generată să poată fi vizualizată în cadrul unui *browser* care suportă *WAP*.

MySQL

În cele ce urmează vom prezenta cele mai utilizate funcții pentru a accesa și interoga baze de date *MySQL* din cadrul *script*-urilor *PHP*.

Funcția mysql_connect

Această funcție realizează conectarea la un *server MySQL*.

Funcția `mysql_connect` primește cinci parametri opționali:

- primul reprezintă numele *server*-ului *MySQL* și dacă lipsește se încearcă conectarea la calculatorul curent.
- cel de-al doilea reprezintă numele utilizatorului și în caz că acesta lipsește se folosește valoarea `root` ca nume de utilizator.
- al treilea parametru reprezintă parola utilizatorului și în caz că acesta lipsește se folosește șirul vid în momentul conectării la *server*.
- al patrulea parametru este de tip logic și are rolul de a indica motorului *PHP* că în cazul în care există o altă conexiune la același *server* să se mai creeze încă una în loc să se reutilizeze cea existentă.
- ultimul parametru este de tip întreg și reprezintă proprietățile pe care le va avea conexiunea. Acest ultim parametru poate fi o combinație a valorilor:
  - `MYSQL_CLIENT_COMPRESS` – indică *server*-ului că va trebui să trimită date comprimate;
  - `MYSQL_CLIENT_IGNORE_SPACE` – indică *server*-ului *MySQL* faptul că va trebui să ignore spațiile care urmează după numele funcțiilor;
  - `MYSQL_CLIENT_INTERACTIVE` – indică *server*-ului *MySQL* să nu închidă conexiunea după scurgerea unui anumit interval de timp în care *script*-ul nu a emis cereri către acesta.

Funcția `mysql_connect` returnează valoarea logică **FALSE** în cazul în care a eșuat conectarea la *server*-ul de baze de date și un identificator de acces la conexiunea către *server*-ul *MySQL* în cazul în care conectarea a reușit.

În continuare prezentăm un exemplu de utilizare a acestei funcții, iar în figura 4 poate fi observat efectul execuției acestui *script*:

```
<HTML>
  <HEAD>
    <TITLE>
      Exemplu mysql_connect
    </TITLE>
  </HEAD>
  <BODY>
  <?PHP
    $conn = mysql_connect("blue", "ginfo",
                          "pginfo");
    if ($conn === FALSE)
      echo "Conectare eșuată.";
    else
      echo "Conexiune stabilită.";
    mysql_close($conn);
  ?>
  </BODY>
</HTML>
```

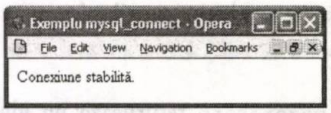


Figura 4: Funcția mysql_connect

Funcția mysql_close

Această funcție primește ca parametru un identificator de acces la o conexiune spre un *server MySQL* și realizează închiderea acesteia.

Am folosit această funcție în exemplul anterior, chiar înainte de terminarea *script*-ului.

Funcția mysql_ping

Funcția `mysql_ping` verifică dacă *server*-ul de *MySQL* a închis conexiunea sau nu. Aceasta nu are nici un parametru și returnează valoarea logică **TRUE** în cazul în care conexiunea nu a fost închisă și valoarea logică **FALSE** în caz contrar.

Funcția mysql_create_db

Această funcție este utilizată pentru a crea o nouă bază de date pe *server*-ul *MySQL*.

Funcția are doi parametri. Primul este de tip șir de caractere și reprezintă numele bazei de date care va fi creată, iar cel de-al doilea parametru reprezintă identificatorul de acces la conexiunea către *server*-ul *MySQL*.

Funcția returnează valoarea logică **TRUE** în cazul în care a reușit să creeze baza de date și valoarea logică **FALSE** în caz contrar.

Funcția mysql_drop_db

Această funcție este folosită pentru a șterge o bază de date.

Funcția `mysql_drop_db` are doi parametri. Primul este de tip șir de caractere și reprezintă numele bazei de date care va fi ștearsă, iar cel de-al doilea parametru reprezintă identificatorul de acces la conexiunea către *server*-ul *MySQL*.

Funcția returnează valoarea logică **TRUE** în cazul în care s-a reușit ștergerea bazei de date și valoarea logică **FALSE** în caz contrar.

Funcția mysql_select_db

Această funcție setează baza de date pentru o conexiune către un *server MySQL* pentru interogările care vor urma.

Funcția `mysql_select_db` are doi parametri. Primul este de tip șir de caractere și reprezintă numele bazei de date care va fi folosită pentru interogările ulterioare, iar cel de-al doilea parametru reprezintă identificatorul de acces la conexiunea către *server*-ul *MySQL*.

Funcția returnează valoarea logică **TRUE** în cazul în care s-a reușit setarea bazei de date și valoarea logică **FALSE** în cazul în care baza de date nu există sau utilizatorul nu are drepturi de acces la aceasta.

Funcția mysql_query

Această funcție se folosește pentru a interoga o anumită bază de date. Funcția `mysql_query` are trei parametri:

- primul este de tip șir de caractere și reprezintă cererea emisă *server*-ului *MySQL*;
- al doilea parametru reprezintă identificatorul de acces al unei conexiuni către *server*;
- parametru este opțional și de tip întreg și reprezintă modul în care va fi returnat rezultatul. Ultimul parametru poate avea valorile `MYSQL_USE_RESULT` sau `MYSQL_STORE_RESULT`.

În cazul în care ultimul parametru lipsește, pentru interogare se va folosi valoarea `MYSQL_STORE_RESULT`, iar rezultatul primit va fi stocat în zone tampon până când va fi utilizat. Dacă, pentru ultimul parametru, se folosește valoarea `MYSQL_USE_RESULT` , atunci rezultatul primit nu va fi stocat, deoarece va fi procesat imediat după apelul funcției.

Utilizarea funcției folosind valoarea `MYSQL_USE_RESULT` pentru ultimul parametru are același efect cu aplicarea funcției `mysql_unbuffered_query` fără a folosi ultimul parametru.

Funcția returnează o valoare de tip *resursă* în cazul în care a fost apelată folosind o interogare care presupune primirea unui rezultat cum sunt comenzile: `SELECT`, `SHOW`, `EXPLAIN` și `DESCRIBE` și nu a apărut nici o eroare, iar în cazul apariției unei erori, este returnată valoarea logică **FALSE**.

În cazul în care funcția `mysql_query` este apelată pentru o interogare care nu presupune primirea unui rezultat, este returnată valoarea logică **TRUE** și valoarea logică **FALSE** în cazul apariției unei erori.

În continuare prezentăm un exemplu de utilizare a acestei funcții, iar în figura 5 poate fi observat efectul execuției acestui *script*:

```
<HTML>
  <HEAD>
    <TITLE>
      Exemplu MySQL
    </TITLE>
  </HEAD>
  <BODY>
  <?PHP
    $conn = mysql_connect("blue", "ginfo",
                                    "pginfo");
    if ($conn === FALSE)
      die("Conectare eșuată.");
    mysql_select_db("ginfo", $conn);
    $res = mysql_query("select * from users
                order by prenume, nume", $conn);
    echo "<TABLE border = 1>";
    echo "<TR><TD align = center>Prenume</TD>";
    echo "<TD align = center>Nume</TD>";
    echo "<TD align = center>Email</TD></TR>";
    while ($lin = mysql_fetch_array($res,
                                MYSQL_BOTH)){
      echo "<TR><TD>".$lin["prenume"]."</TD>";
      echo "<TD>".$lin["nume"]."</TD>";
      echo "<TD><A href = \"mailto:".
                          $lin["email"]."\">";
```

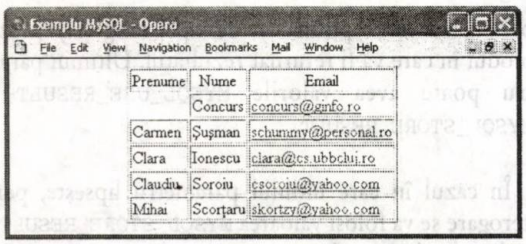


Figura 5: Exemplu de utilizare MySQL

```
        echo $lin["email"]."</A></TD></TR>";
    }
    echo "</TABLE>";
    mysql_free_result($res);
    mysql_close($conn);
  ?>
  </BODY>
</HTML>
```

Funcţia mysql_unbuffered_query

Această funcţie se foloseşte pentru a interoga o anumită bază de date şi are aceeaşi parametri cu funcţia anterioară.

Diferenţa dintre această funcţie şi `mysql_query` este aceea că, în cazul funcţiei `mysql_unbuffered_query`, dacă ultimul parametru lipseşte, se foloseşte la interogarea bazei de date valoarea `MYSQL_USE_RESULT`.

Funcţia mysql_real_escape_string

Această funcţie este utilă în momentul în care se doreşte introducerea în cadrul unei interogări a unui şir de caractere care nu poate fi interpretat corect de către *server*-ul *MySQL* şi realizează transformarea şirului într-unul care poate fi interpretat.

Funcţia `mysql_real_escape_string` are doi parametri. Primul este de tip şir de caractere şi reprezintă şirul care va fi transformat, iar cel de-al doilea reprezintă identificatorul de acces la o conexiune către un *server MySQL*.

Funcţia returnează un şir de caractere care poate fi interpretat corect de către *server*-ul *MySQL*.

De exemplu, dacă apelăm această funcţie pentru şirul de caractere `D'apoi` vom obţine rezultatul `D\'apoi`, deoarece caracterul `"'"` (apostrof) are o altă semnificaţie pentru *server*-ul *MySQL* dacă nu este precedat de caracterul `"\"`.

Funcţia mysql_free_result

Această funcţie se foloseşte pentru a elibera memoria alocată stocării unui rezultat primit în urma unei interogări de la un *server MySQL*.

Funcţia `mysql_free_result` are un singur parametru, şi anume o variabilă de tip *resursă* a cărei valoare a fost setată folosind una dintre funcţiile `mysql_query` sau `mysql_unbuffered_query`.

Funcţia returnează valoarea logică **`TRUE`** în cazul în care a reuşit să elibereze memoria utilizată de variabila de tip *resursă* primită ca parametru şi valoarea logică **`FALSE`** în caz contrar.

Funcţia mysql_num_rows

Această funcţie returnează numărul de înregistrări conţinute de către un rezultat primit de la *server*-ul *MySQL*.

Funcţia `mysql_num_rows` are un singur parametru, şi anume o variabilă de tipul *resursă* a cărei valoare a fost setată folosind una dintre funcţiile `mysql_query` sau `mysql_unbuffered_query`.

Funcţia returnează un număr întreg care reprezintă numărul de înregistrări conţinute de variabila de tip *resursă* primită ca parametru.

În continuare prezentăm un exemplu de utilizare a acestei funcţii, iar în figura 6 poate fi observat efectul execuţiei acestui *script*:

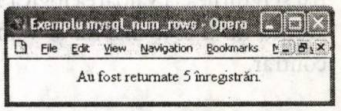


Figura 6: Funcţia mysql_num_rows

```
<HTML>
  <HEAD>
    <TITLE>
      Exemplu mysql_num_rows
    </TITLE>
  </HEAD>
  <BODY>
  <?PHP
    $conn = mysql_connect("blue", "ginfo",
                                    "pginfo");
    if ($conn === FALSE)
      die("Conectare eşuată.");
    mysql_select_db("ginfo", $conn);
    $res = mysql_query("select * from users
              order by prenume, nume", $conn);
    $num_rows = mysql_num_rows($res);
    echo "Au fost returnate $num_rows
                                înregistrări.";
    mysql_free_result($res);
    mysql_close($conn);
  ?>
  </BODY>
</HTML>
```

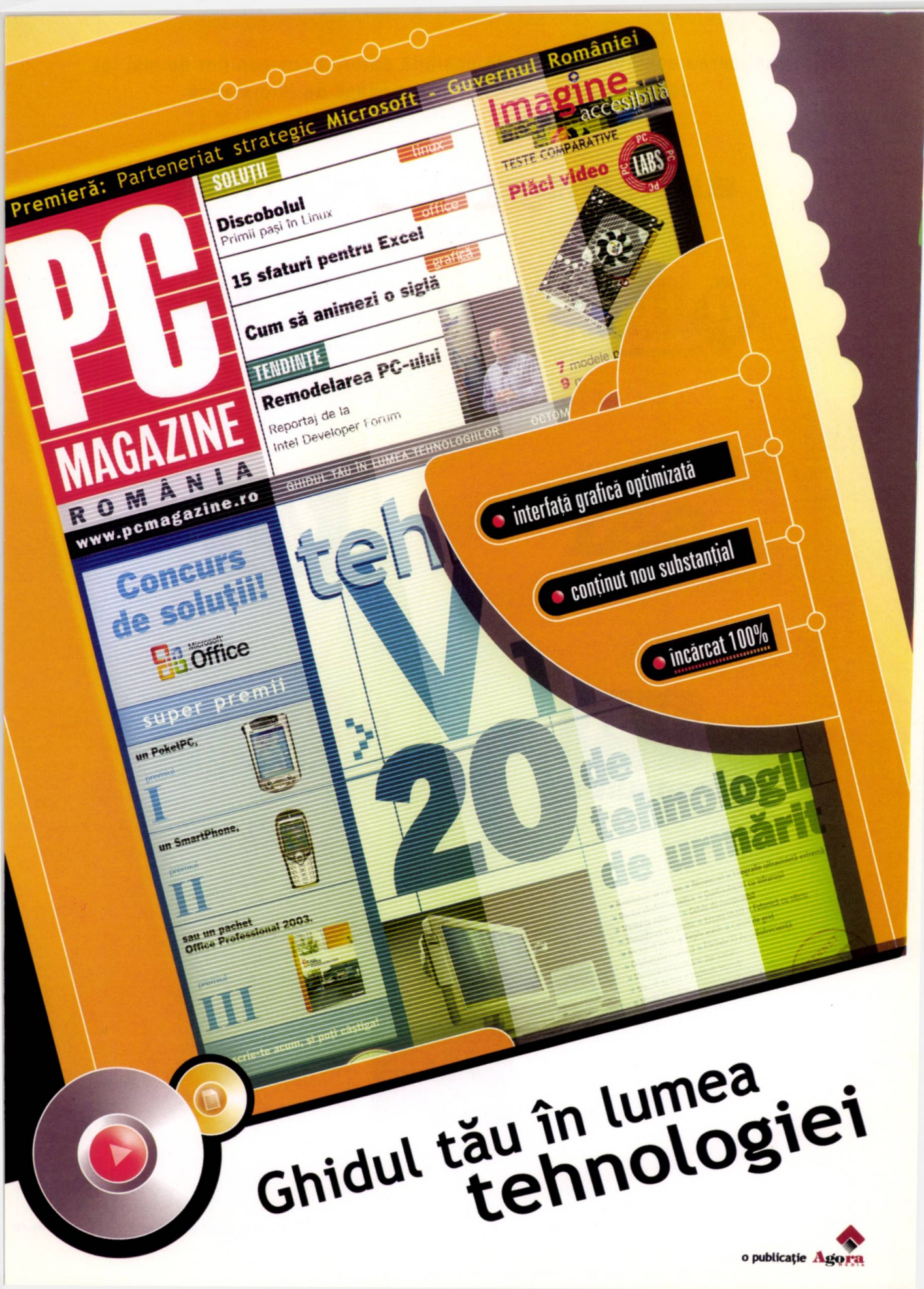
Funcţia mysql_info

Această funcţie returnează un şir de caractere care conţine informaţii referitoare la ultima interogare a unei baze de date în urma căreia nu s-a primit nici un rezultat, cum este cazul funcţiilor `INSERT` sau `UPDATE`.

Funcţia `mysql_info` are un singur parametru care reprezintă identificatorul de acces al unei conexiuni spre un *server MySQL*.

Va urma...

În numărul următor vom continua prezentarea funcţiilor referitoare la accesarea bazelor de date *MySQL* şi vom începe prezentarea funcţiilor de poştă electronică folosite de programatorii *web* pentru a-i anunţa pe eventualii utilizatori ai *site*-ului lor de apariţia unor evenimente.


Premieră: Parteneriat strategic Microsoft - Guvernul României
PC MAGAZINE ROMÂNIA
www.pcmagazine.ro
SOLUȚII
Discobolul
Primii pași în Linux
15 sfaturi pentru Excel
Cum să animezi o sigiă
TENDINȚE
Remodelarea PC-ului
Reportaj de la Intel Developer Forum
Imagine accesibilă
TESTE COMPARATIVE
Plăci video
PC LABS
GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR
Concurs de soluții!
Office
super premii
un PocketPC,
un SmartPhone,
sau un pachet Office Professional 2003,
20
interfață grafică optimizată
conținut nou substanțial
încărcat 100%
Ghidul tău în lumea tehnologiei
o publicație Agora

Publicarea acestei reviste este posibilă datorită companiilor de mai jos care sprijină învățământul românesc de informatică

SPONSORI DE PLATINĂ

FORTE COMPUTERS
Flexibility. Made in Romania

IBM®

s&t
IT SOLUTIONS & SERVICES

hp®

net consulting

FUJITSU COMPUTERS SIEMENS

SPONSOR DE AUR

CCT
CABINETUL CĂLIN TATOMIR

Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-166.516

Gifo
Februarie 2004
Vol. 14/2
Lei 30.000

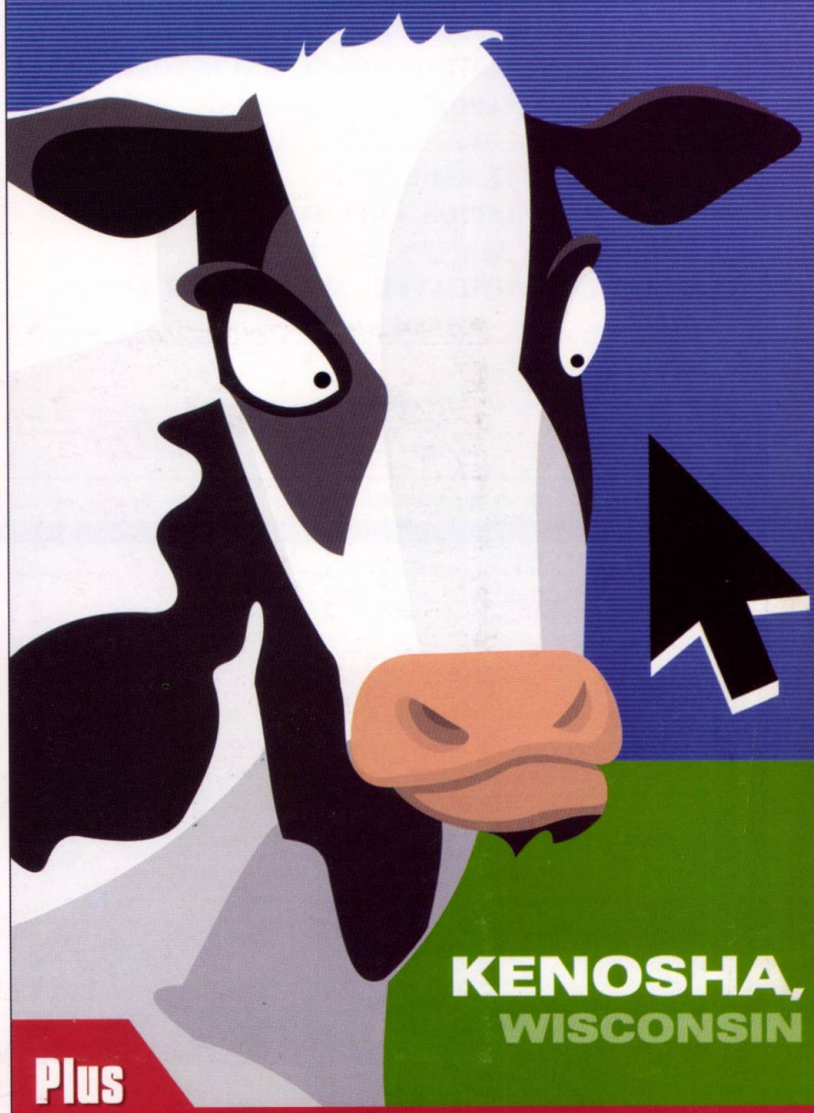
Gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

IOI 2003

Soluțiile problemelor propuse



KENOSHA, WISCONSIN

Plus

actual
În căutarea unei cafele perfecte

babel
Testarea și depanarea programelor

probleme
Olimpiada LOCALĂ de Informatică

cantitate (bucăți)

TALON DE COMANDĂ

Corespondență Răspuns CR Se taxează la destinație

- ☐ **INIȚIERE ÎN DESIGN** de Robin Williams. Preț: 125 000* lei
- ☐ **MICROSOFT OFFICE XP PENTRU WINDOWS GHID DE ÎNVĂȚARE RAPIDĂ PRIN IMAGINI** de Steve Sagman. Preț: 249 000* lei
- ☐ **DESIGN PENTRU WEB** de Robin Williams, John Tollet. Preț: 299 000* lei
- ☐ **INIȚIERE ÎN WINDOWS XP GHID DE PREZENTARE A SISTEMULUI DE OPERARE** de Larry Magid. Preț: 225 000* lei
- ☐ **QUARKXPRESS 4 și 5 SHORTCUTS PENTRU PC ȘI MACINTOSH** de David Blatner. Preț: 120 000* lei
- ☐ **CALCULATOARE ȘI INTERNET. DICȚIONAR EXPLICATIV** de Random House Webster's. Preț: 140 000* lei
- ☐ **PHOTOSHOP 7 PENTRU WINDOWS ȘI MACINTOSH GHID DE ÎNVĂȚARE RAPIDĂ PRIN IMAGINI** de Elaine Weinmann, Peter Lourekas. Preț: 350 000* lei
- ☐ **HTML PENTRU WORLD WIDE WEB GHID DE ÎNVĂȚARE RAPIDĂ PRIN IMAGINI** de Elizabeth Castro. Preț: 375 000* lei
- ☐ **MACROMEDIA® DREAMWEAVER® MX ÎN 24 DE LECȚII** de Betsy Bruce. Preț: 325 000* lei

* fără TVA

EXPEDITOR:

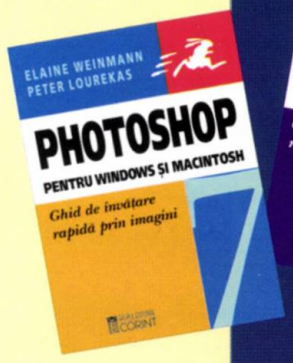
Nume: ..... Prenume: .....
Str. ..... Nr. ..... Bl. ..... Sc. ..... Et. ..... Ap. .....
Localitate ..... Cod poștal ..... Jud. / Sect. ..... Tel. .....

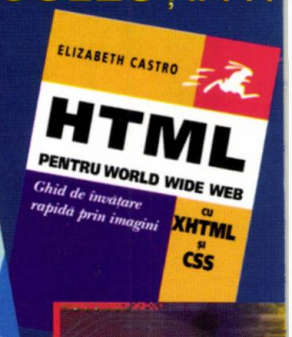
Completați talonul de comandă conform opțiunilor dumneavoastră și introduceți-l – netimbrat – în orice cutie poștală.

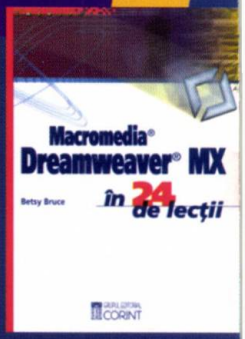
DESTINATAR:
Editura CORINT,
CP 1-591,
București
tel.: 223.19.28
222.19.49
tel/fax: 222.40.34

GRUPUL EDITORIAL CORINT

Ultimele apariții în COLECȚIA IT







Cine are carte, are parte!

... de înțelegerea rețelelor!

Cea mai bună introducere din lume în rețele de calculatoare
- complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o reducere semnificativă!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65; 0721-927.499

Vol. 14, Nr. 2 /2004

gazeta de informatică

Sumar

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Director Bursele Agora:
Mihai Scorțaru
Redactor-șef:
Claudiu Soroiu
Redactor-șef adjunct:
Carmen Șușman
Redactori:
prof. *Eugen Ionescu*
Mihai Stroe
Colaboratori:
prof. *Lucian Ilea*
Bogdan Mărincaș
Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
400091, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/592.422, 0265/266.516
Fax: 0265/266.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359

Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu
Denis Marioara
Marius Comărniceanu
Difuzare:
Florin Dinu
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134

Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

actual.. PAG. 4

Căutând ADEVĂRUL, navigând spre... JAVA
Clara IONESCU
~ în căutarea unei ~
~ cafele perfecte ~

probleme.. PAG. 6

Olimpiada LOCALĂ de Informatică
~ enunțurile problemelor~
~ propuse spre rezolvare ~
~ la Cluj-Napoca și Tg. Jiu ~

soluții.. PAG. 17

IOI 2003
Mihai STROE

Olimpiada de informatică a EUROPEI CENTRALE
~ soluțiile problemelor ~
~ propuse spre rezolvare ~
~ la cele două concursuri ~

concurs.. PAG. 26

Despre rundele #06-#15
~ soluțiile problemelor ~
Top 50 după 15 runde
Rundele #16-#25
~ enunțurile problemelor ~
~ propuse spre rezolvare ~

focus.. PAG. 35

To be XOR not to be
Lucian ILEA
Bogdan Mărincaș
~ utilizarea operatorului ~
~ binar XOR ~

babel.. PAG. 36

TESTAREA și DEPANAREA programelor
Mihai STROE
~ indicații prețioase ~
~ pentru concursuri ~

techno.. PAG. 41

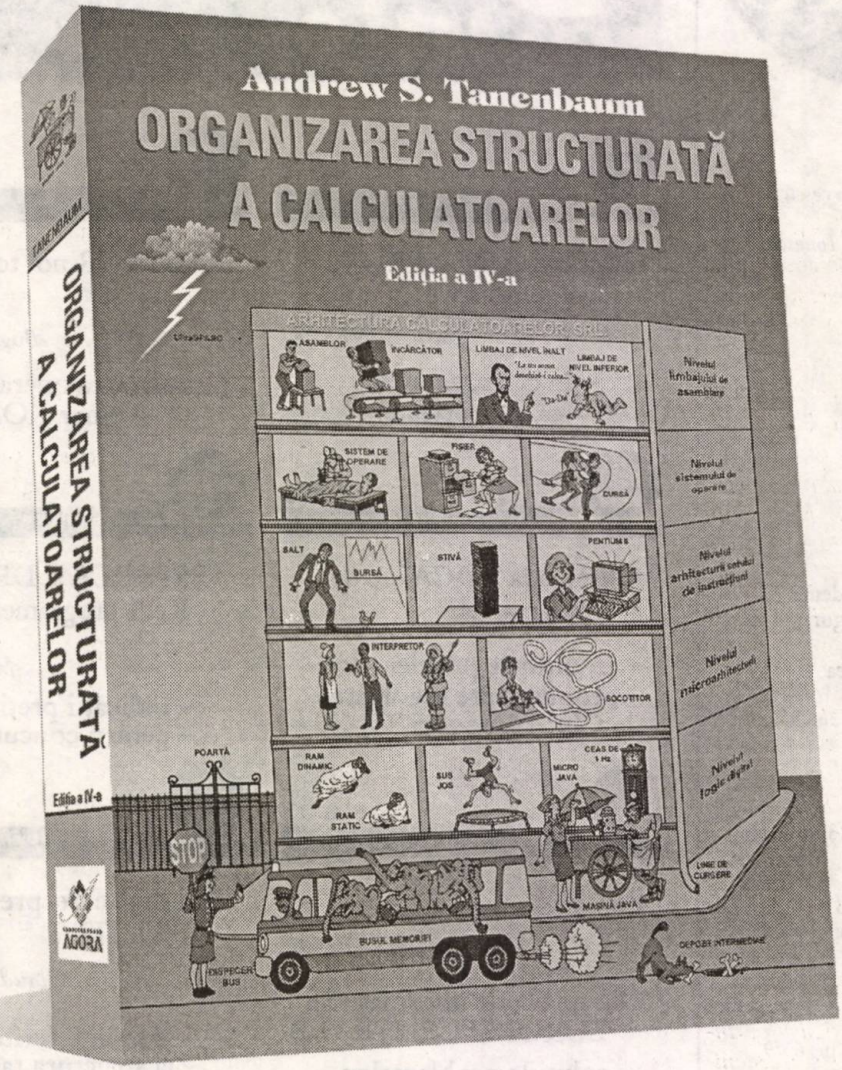
Mai multe despre PARTIȚII
Claudiu SOROIU
~ detalii referitoare ~
~ la structura tabelei ~
~ de partiții ~

internet.. PAG. 44

Pagini WEB cu PHP 4
Mihai SCORȚARU
Claudiu SOROIU
~ accesarea bazelor ~
~ de date MySQL și ~
~ transmiterea mesajelor ~
~ electronice ~

Cine are carte, are parte!

...de înțelegerea calculatoarelor!



Comandă-o acum și dacă ești student

primești și o **REDUCERE** semnificativă!

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65, 0721-927.499

Mai e O ȘANSĂ!

Motto: Leneșul mai mult aleargă...

Dragi cititori, anul 2004 este unul dedicat, fie că vrem, fie că nu vrem, campaniei electorale... Eu însă am un plan mult mai mic, dar de viitor, și căruia îi voi dedica doar o pagină. Restul nu mai depinde de mine!

Cu permisiunea dumneavoastră vă voi spune o scurtă poveste. Hazlie.

A fost odată ca niciodată un elev care se chema Cu Minte. Nu era un elev eminent, nici chiulangiu. Cu Minte mergea la școală zi de zi cu speranțe care toate începeau cu "Doamne dă... : să nu mă asculte azi la nici o materie! să nu mă scoată la tablă la mate! să nu îmi verifice temele! să pot copia tot în caz că ne dă exteu'! să fie bolnav(ă) profu' (profa) Icsulescu, respectiv Icsuleasca, să plecăm mai repede acasă! să mă îmbolnăvesc mâine și să nu mai pot veni la școală, dar să nu fie o boală prea grea!" etc. Cu siguranță știți voi și de ce le avea și să încheiați lista de dorințe a lui Cu Minte! Nu-i așa :)?

Ei bine, în ziua respectivă Cu Minte și-a văzut toate dorințele îndeplinite. A doua zi la fel. A treia zi la fel. A patra zi la fel. A cincea zi la fel. Am uitat să vă spun că, de fiecare dată când scăpa, Cu Minte spunea: "Doamne, mulțumesc că m-ai ajutat! Dacă mă mai ajuți, promit că de mâine mă apuc de învățat!" Dar, bineînțeles că niciodată nu se ținea de cuvânt.

Și, după mulți ani grei de "studiu", Cu Minte a reușit să termine școala și să promoveze examenele cu bine, dus pe brațe de colegii lui care mereu i-au șoptit când era ascultat, l-au lăsat să copieze la lucrări de control, l-au salvat din situații critice, adică l-au «ajutat». Se pare că acest Cu Minte a fost și destul de norocos pentru că ajungea aproape tot timpul printre primii.

Mare a fost bucuria părinților când și-au văzut fiul om cu multă carte!

Acum Cu Minte este mare director la o firmă. Tăticul și-a lăsat odrasla școlită în locul lui. Și atenție! Firma merge foarte bine. Norocul mare este că acest Cu Minte nu trebuie decât să semneze acte. Și trebuie să recunosc faptul că o face foarte bine și chiar îi place!

Sfârșit de poveste!

Se poate observa la tot pasul că unul sau mai mulți indivizi, pe care îi voi numi în continuare cu titlul generic "Cu Minte", sunt *sateliții* unei persoane deștepte, inteligente și conștiincioase, pe care acesta o alintă cu sintagma: "ești un adevărat prieten!". Până aici, toate sunt bune și frumoase! Dar gândește-te, "prietene adevărat", s-ar putea ca într-o zi să ajungi să lucrezi pentru Cu Minte... Să te milogești la el pentru o mărire de salariu pe care este cert că o meriți.

Ei, te-am trezit la realitate? Doare, nu? Deocamdată cel puțin teoretic.

Spune, te rog, tu câți "Cu Minte" ai în clasă? Doar atâția? Mai caută, mai caută...

Și acum voi deschide o "discuție" despre solidaritea dintre elevi.

În general elevii sunt mai solidari între ei la rău... Vă dau un exemplu. Într-o clasă se sparge un geam. Dirigintele întreabă: "Cine a spart geamul?" În prima fază nimeni nu zice nimic. Dirigintele insistă: "Ziceți cine a spart geamul, că doar nu s-a spart singur..." Elevii prind curaj: "Nu știm!" și se miră cu niște fețe de "habar nu avem cine a avut tupeul să vină la noi în clasă și să ne spargă geamul". Culmea este că nici unul nu se dă de gol. Dirigintele, bietul, încearcă să îi convingă: "Dacă îmi spuneți, vă promit că nu vă fac nimic!" De fapt, "încearcă să îi convingă" este frumos spus. El se roagă de ei! Nasol! Elevii nu vor să spună! Și punct. În ziua următoare dirigintele iar vine să se roage de ei. Surpriză! Geam nou montat deja. De elevi. Dirigintele: "Cine a pus geamul?" Elevii: "Nu știm, probabil că cel care l-a spart. Dar haideți, domn' diriginte să încheiem cu povestea asta. Avem geam nou! Ce mai contează cine l-a spart. Probabil a fost din greșeală!"

Păi, dragii mei, contează! Și vă dau un singur motiv. Dacă te duce capul la trăznăi, trebuie să te ducă și să îți asumi răspunderea...

În acest exemplu trebuie să îți asumi răspunderea față de cineva. Dar în povestire mea, da, trebuie să răspunzi în fața ta. E mai greu! Mai bine iei atitudine acum pentru a nu te da mai târziu cu capul de pereți!

Mai bine îi spui frumos lui Cu Minte să se pună cu burta pe carte și că o să îl ajuți dacă nu înțelege ceva. Asta înseamnă să fii solidar și corect! Într-o zi îți va mulțumi.

Iar dacă ești chiar tu Cu Minte, atunci pune-te pe treabă, nu ezita și nu-ți fie rușine să ceri ajutorul colegilor și nu uita ca, imediat ce obții niște rezultate, să îți schimbi numele!

Carmen Șușman

În căutarea ADEVĂRULUI, navigând spre... JAVA

Clara Ionescu

Suntem printre puținele țări din lume, unde printre disciplinele pe care le studiază un elev în liceu (și chiar și în gimnaziu) se află și *informatica*. Există locuri, unde învățământul preuniversitar oferă cunoștințe din domeniul *Tehnologiei Informației*, dar în foarte puține școli învață elevii algoritmi și programare. În România acest învățământ își are începuturile în anul școlar 1971-1972 și de atunci a suferit multe modificări, din păcate trecând de mult de apogeu. Ce ar trebui să învețe adolescenții interesați de informatică în secolul XXI? O întrebare la care se va mai căuta răspunsul mult timp...

"O cafea bună..." scrie *Mihai Scorțaru* în nr. 13/7 al *Gazetei de Informatică* și se lansează într-o pledoarie în favoarea limbajului *Java*. Cred că *Java* nu (prea) are nevoie de avocați, totuși înțeleg entuziasmul. Este normal să îmbrățișezi ultima "jucărie" și să le uiți în colțuri pe cele mai vechi. Sunt de acord cu subpunctele ordonat expuse în care se discută avantejele acestui mediu care, evident nu întâmplător a apărut în lumea informaticii. Dar, atunci când discutăm despre programa școlară de informatică din licee, pe lângă aceste avantaje trebuie să luăm în considerare și o serie de alte aspecte.

În cele ce urmează voi lua pe rând câteva dintre afirmații, încercând apoi să îmi sintetizez concluziile. Nu mă mândresc cu ele, mă suspectez că și pe mine mă domină un fel de comoditate, chit că de când mă știu, ca mulți colegi profesori de informatică, învăț mereu.

Încep cu un citat care mă necăjește: "*...studentul **român** nu învață nimic la facultate. Aproape toți programatorii susțin că au învățat singuri...*" (evidențierea îmi aparține). Las la o parte faptul că de aici se înțelege că alți studenți (care nu sunt ***români***) ar învăța mai mult la facultate. Nu voi folosi argumente cum ar fi: programatorii amintiți exagerează.

De asemenea, cred cu încăpățânare că studenții - indiferent de cetățenie - învață *dacă vor* și nu învață dacă le trebuie doar o hârtie din care să rezulte că au trecut printr-o instituție de învățământ superior.

Problema este una eternă. Eu nu sunt programator, dar și în cazul meu s-a adeverit că ceea ce am învățat la facultate fie am uitat foarte repede, fie s-a dovedit a fi depășit într-un timp relativ scurt. Și totuși, nu susțin că am absolvit facultatea degeaba. Ceea ce am învățat, de fapt este o "atitudine" față de știință, o curiozitate, o deprindere în a căuta și a selecta informația, o maturitate în a formula întrebări și de a finaliza aplicații. Chiar și faptul că am învățat multă matematică, azi se reflectă în preocupările mele, cu toate că nu sunt matematician.

În continuare citez din nou din articol: "*După ce termină liceul își dă seama (elevul doritor să devină programator) că nu știe nimic din ceea ce are nevoie.*" Dacă am aștepta de la liceu să-l învețe pe fiecare tânăr ceea ce acesta "are nevoie", ne-ar trebui atâtea școli câți elevi... Fiecare vrea să devină altceva, fiecare va avea nevoie în viitor de altceva.

Îi reamintesc prietenului și fostului meu elev *Mihai Scorțaru* că școlile sunt pentru grupuri de elevi, care evident sunt foarte diferiți, iar învățământul preuniversitar are o serie de *obiective generale* care sunt aceleași la noi și în orice altă țară. Aceste obiective sunt la rândul lor descompuse și concretizate pe discipline. Printre obiectivele generale ale disciplinei *Informatică* nu apare și nici nu trebuie să apară obiectivul: "*să devină programator*". Liceul trebuie să deschidă multe "ferestre", să ofere cunoștințe de cultură generală (începând cu literatura și terminând cu... informatica sau invers). Liceul nu oferă o specializare precisă și exclusivă, doar dacă avem de a face cu unul *vocațional*. În plus, știm că nu toți elevii din clasele de matematică-informatică *doresc* sau *pot* să devină programatori.

Aici consider că am ajuns de fapt la "nodul Gordian" al problemei. Respectiv, dacă în *România* ar exista câteva (5-

6) astfel de licee (ca în anii '70), atunci aș fi chiar prima care m-aș alinia cu *Mihai Scorțaru* și pledoaria lui pentru *Java*. Dar situația reală a învățământului preuniversitar de informatică românesc este alta. Un număr foarte mare de elevi "nimeresc" în clase de matematică-informatică, iar majoritatea lor nu știu nimic despre "info", încă nu știu ce vor să fie după terminarea liceului. Și atunci, liceul trebuie să-și îndeplinească sarcina fundamentală, formulată în obiectivele generale ale fiecărei discipline.

Să vedem un astfel de obiectiv: "*elevii își vor forma o gândire logică, disciplinată*" (în unele locuri se precizează chiar "*algoritmică*"). Am convingerea că acest obiect se numără printre cele mai importante și totodată printre cele mai greu de realizat.

Îmi imaginez că partenerul meu de discuție dă din cap și concluzionează: "*de acord, dar elevul la fel de bine își poate forma acest tip de gândire și dacă aplicațiile le scrie într-un limbaj de programare nou, performant și nu neapărat într-unul învechit*"...

Da, aparent ar avea dreptate. De ce doar *aparent*? Pentru că, de exemplu, datorită faptului că devine obligatoriu învățământul de 9 sau 10 ani, în clasa a IX-a există o singură oră de informatică. Știu, unii sunt de părere că nu este bine, că trebuie schimbat. Da, ar fi de dorit, dar nu pentru întreaga suflare de copii de clasa a IX-a din clasele de matematică-informatică existente azi.

Revin cu "ideea fixă": ar trebui să existe în *România* un învățământ de informatică *vocațional*. Acolo, în 6-8 ore pe săptămână ar putea să învețe, elevii cu adevărat doritori să devină programatori, multe lucruri interesante, multe lucruri noi. Acolo, profesorii ar avea și ei "succes școlar" și s-ar pregăti pentru fiecare oră cu entuziasm. Până când majoritatea elevilor nu doresc să devină programatori, acest lucru este inutil. Rămân obiectivele generale: elevii trebuie să-și formeze deprinderea *de a gândi cu creierele lor proprii, de a rezolva probleme care necesită gândire*, nu doar să utilizeze componente de software, *gândite* de alții. Se știe că mulți "programatori", de regulă, nu fac altceva decât să implementeze ceea ce o *altă persoană* a proiectat: structuri de date, algoritmi. Cu siguranță știți cine este mai apreciat...

Mult huiduitul limbaj *Pascal* a fost creat într-adevăr de mult timp. Dar a fost creat cu scopul clar de a fi folosit în învățământ. Cei care l-au proiectat, au dorit să ofere un instrument simplu, "disciplinat" care să-l ajute pe începător *pentru a face primii pași în programare*. Și din nou îmi imaginez că interlocutorul încearcă să mă convingă că și *Java* este un limbaj disciplinat, poate chiar *mai* disciplinat decât *Pascalul*. Îi dau dreptate. Programarea orientată obiect protejează datele, chestiune rezolvată în *Java* într-un mod elegant și performant. Dar, elevii ar fi obligați să învețe o serie de cunoștințe, o mulțime de secrete ale "meseriei", mult mai multe decât în cazul *Pascalului*. Ar avea la dispoziție o bibliotecă imensă de subprograme, din care greu ar învăța să selecteze ceea ce au nevoie. Ar ajunge în situația asemănătoare cu cel care trage cu tunul într-o muscă...

Revin la un alt citat: "*Cineva îi va învăța (pe elevi) să deseneze interfețe în Delphi și vor deveni programatori.*" Oare doar atât lipsește ca un elev să devină programator? Atunci nu văd nici o problemă... "*Mi se pare inutil acest efort de a îi învăța pe elevi algoritmică... E necesară, într-adevăr, dar nu atât de necesară precum cunoașterea unui limbaj de programare modern.*" Nu sunt de acord din capul locului. În rest: *Java* este modern - astăzi. Dar problema este veșnică. Mâine *Java* va fi unul "depășit". Și uite așa. Și abecedarul este "vechi", dar toți copiii au nevoie de el, indiferent în ce limbă vor citi, scrie și vorbi.

Subiectul discuției noastre are implicații și în învățământul universitar. Într-adevăr, acei absolvenți care doresc "să devină programatori" se înscriu la o facultate unde urmează cursuri universitare de informatică. Dar acolo vor avea colegi care nu au absolvit o clasă de matematică-informatică. Aceștia urmează să fie introduși mai întâi în cunoștințe elementare și până când vom avea cursuri separate pentru *începători* și pentru *avansați* va mai trece mult timp.

În final voi comenta pe scurt "tonul certăreț" al articolului în discuție privind salariile profesorilor. Subiectul a devenit agasant și personal m-am săturat de el. Consider că dacă un profesor merge la școală (la "serviciu") pentru salariu, atunci ar fi mai bine să nu meargă.

Încerc să trag concluziile: dacă am avea școli de informatică *vocaționale*, s-ar justifica învățarea unor medii de programare performante. Dar într-un mediu școlar unde, acum câțiva ani, am fost criticată de părinți din cauza că folosesc cuvinte sofisticate, precum "*entitate*", unde trebuie ore de explicații până când elevii înțeleg diferența dintre *o funcție și o procedură, dintre parametri transmiși prin valoare și prin referință* etc. cred că trebuie să ne mulțumim cu mai puțin.

Mihai Scorțaru a fost olimpic, apoi student eminent. Acum programează cu ușurință în *Java*. Bravo lui! Dar a crede că un învățământ preuniversitar de stat *ar trebui* și *ar putea* să convingă generații de elevi să învețe să programeze într-un limbaj de programare îmbrățișat de lumea programatorilor profesioniști de *azi*, consider că este o exagerare.

Am covingerea că nu *nu limbajele de programare rezolvă problemele ci persoanele*. Pentru ca cineva să devină scriitor apreciat, nu este suficient să posede un vocabular bogat. Trebuie să aibă ceva de spus.

Clara Ionescu este cadru didactic la Universitatea Babeș-Bolyai din Cluj-Napoca și poate fi contactată prin e-mail la adresa clara@cs.ubbcluj.ro

24 ianuarie 2004, Cluj-Napoca

Olimpiada LOCALĂ de Informatică

Pe data de 24 ianuarie 2004 a avut loc, la Cluj-Napoca, ediția din acest an a fazei locale a olimpiadei de informatică. La acest concurs au participat elevi din clasele V - XII. Elevii din clasele V - VIII s-au confruntat cu câte două probleme, iar cei din clasele IX - XII cu câte trei. În continuare vă prezentăm enunțurile celor 17 probleme propuse spre rezolvare.

Clasa a V-a

P020401: Mingi

Andrei are *n* mingi. Pe fiecare minge este scris un număr care conține cel mult trei cifre.

Sarcina voastră este să scrieți un program care să-l ajute pe *Andrei* să numere câte cifre impare sunt în total în numerele scrise pe mingi.

Date de intrare

De la *intrarea standard* (tastatură) se va citi de pe prima linie numărul *n*. De pe următoarele *n* linii se vor citi numerele înscrise pe mingi.

Date de ieșire

La *ieșirea standard* (ecran) se va afișa un singur număr natural care reprezintă numărul total de cifre impare înscrise pe mingi.

Exemplu

Intrare standard

```
3
346
6
145
```

Ieșire standard

```
3
```

Explicație

Pe prima minge se află o singură cifră impară, și anume 3. Pe a doua minge nu se află nici o cifră impară, iar pe cea de-a treia minge se află cifrele impare 1 și 5.

În concluzie, pe cele trei mingi sunt înscrise trei cifre impare.

P020402: Case

Bunicii lui *Mihai* i-au promis acestuia o prăjitură dacă rezolvă următoarea problemă: *Pe strada bunicilor casele sunt numerotate de la 1 la n și sunt dispuse astfel: cele cu numere pare pe partea stângă și cele cu numere impare pe partea dreaptă. Să se determine suma numerelor caselor de pe fiecare parte a străzii.*

Scrieți un program care să îl ajute pe *Mihai* să obțină prăjitura.

Date de intrare

De la *intrarea standard* (tastatură) se va citi de pe prima linie numărul *n*, care reprezintă numărul caselor de pe ambele părți ale străzii pe care locuiesc bunicii lui *Mihai*.

Date de ieșire

La *ieșirea standard* (ecran), pe prima linie se va afișa un singur număr natural care reprezintă suma numerelor caselor care se află pe partea stângă a străzii pe care se locuiesc bunicii lui *Mihai*, iar pe cea de-a doua linie se va afișa un singur număr natural care reprezintă suma numerelor caselor care se află pe partea dreaptă a străzii.

Exemplu

Intrare standard

```
7
```

Ieșire standard

```
12
16
```

Explicație
Pe partea stângă se află casele având numerele 2, 4 și 6, a căror sumă este 12, iar pe partea dreaptă se află casele având numerele 1, 3, 5 și 7, a căror sumă este 16.

Clasa a VI-a

P020403: Săniuș

La un concurs de săniuș au rămas în cursă, după preliminarii, trei copii.

Ordinea intrării în finală este dată de greutatea totală a fiecărui copil, greutate dată de suma dintre greutatea copilului și greutatea saniei. Cel cu greutatea totală cea mai mare va intra primul în concurs, iar cel cu greutatea totală cea mai mică va intra ultimul.

Sarcina voastră este să stabiliți ordinea intrării în concurs a celor trei copii și premiul obținut de câștigător știind că premiul constă într-un număr de prăjituri egal cu diferența dintre greutatea totală maximă și greutatea totală minimă.

Date de intrare
De la *intrarea standard* (tastatură) se va citi, de pe prima linie, un singur număr natural care reprezintă greutatea primului copil, iar de pe a doua linie se va citi greutatea saniei primului copil.

De pe a treia linie se va citi greutatea celui de-al doilea copil, iar de pe a patra linie se va citi greutatea saniei celui de-al doilea copil.

De pe următoarea linie se va citi greutatea celui de-al treilea copil, iar de pe ultima linie (a șasea) se va citi greutatea saniei celui de-al treilea copil.

Date de ieșire
La *ieșirea standard* (ecran), pe prima linie se va afișa numărul de ordine al copilului care intră primul în concurs, pe cea de-a doua linie se va afișa numărul de ordine al copilului care intră al doilea în concurs, iar pe cea de-a treia linie se va afișa numărul de ordine al copilului care intră al treilea în concurs. Pe cea de-a patra linie se va afișa numărul de prăjituri pe care îl va primi câștigătorul.

Exemplu
Intrare standard
```
34
5
40
7
38
6
```

Ieșire standard
```
2
3
1
8
```

Explicație
Suma greutăților primului copil și a saniei sale este 39, suma greutăților celui de-al doilea copil și a saniei sale este 47, iar suma greutăților celui de-al treilea copil și a saniei sale este 44.

Deci, ordinea intrării în concurs este 2, 3, 1 și premiul constă în 47 - 39 = 8 prăjituri.

P020404: Numere subunitare

Se consideră un șir de n numere reale. Se cere să se calculeze procentul de numere reale pozitive subunitare conținute în acest șir.

Date de intrare
De la *intrarea standard* (tastatură) se va citi de pe prima linie un singur număr natural n care reprezintă numărul de numere care se află în șir.

De pe cea de-a doua linie se vor citi n numere reale, care reprezintă valorile din șir.

Date de ieșire
La *ieșirea standard* (ecran) se va afișa procentul de numere reale pozitive subunitare conținute în șirul citit. Numărul va fi scris cu o precizie de două zecimale exacte.

Exemplu
Intrare standard
```
8
2.5 -2.17 0.23 1.04 0.045 -0.13 0.204 5.37
```

Ieșire standard
```
37.50
```

Explicație
Dintre cele opt numere ale șirului, trei sunt numere pozitive subunitare, și anume 0.23, 0.045 și 0.204.

Așadar, procentul cerut în enunțul problemei este obținut astfel: 3 / 8 · 100 = 37.5.

Clasa a VII-a

P020405: Pacienți

Într-un spital sunt internați n pacienți. Pentru pacienți se cunosc valorile c_1, c_2, c_3, ..., c_n care reprezintă valoarea colesterolului lor.

Pacienții sunt sănătoși dacă au colesterolul cuprins între limitele a două valori a și b, unde $a < b$.

Să se determine numărul pacienților sănătoși și media aritmetică a colesterolului pacienților bolnavi, dacă există, astfel de pacienți.

Date de intrare
De la *intrarea standard* (tastatură) se va citi de pe prima linie numărul n, care reprezintă numărul pacienților.

De pe următoarea linie se vor citi valorile c_1, c_2, c_3, ..., c_n, care reprezintă valorile colesterolului pacienților.

De pe cea de-a treia linie se vor citi două valori *a* și *b*, care reprezintă limitele între care trebuie să se afle colesterolul unui pacient pentru ca acesta să fie considerat sănătos.

Date de ieșire

La *ieșirea standard* (ecran), pe prima linie se va afișa un singur număr natural care reprezintă numărul pacienților sănătoși.

Pe cea de-a doua linie se va afișa media aritmetică a valorilor colesterolului pacienților bolnavi sau valoarea -1 dacă nu există astfel de pacienți.

Exemplu

Intrare standard

```
5
100 130 340 400 200
80 300
```

Ieșire standard

```
3
370
```

Explicație

Dintre cei cinci pacienți trei au valoarea colesterolului cuprinsă între 80 și 300, iar doi sunt bolnavi (colesterolul lor este 340, respectiv 400).

Media aritmetică a valorilor colesterolului pentru cei doi pacienți bolnavi este 370.

P020406: Numere prime

Se consideră un număr natural *n*. Se cere să se determine primele *n* numere prime.

Date de intrare

De la *intrarea standard* (tastatură) se va citi de pe prima linie numărul *n*, care reprezintă numărul de numere prime care trebuie determinate.

Date de ieșire

La *ieșirea standard* (ecran) se vor afișa pe o singură linie cele *n* numere determinate, separate printr-un singur spațiu. Numerele se vor scrie în ordine crescătoare.

Exemplu

Intrare standard

```
13
```

Ieșire standard

```
2 3 5 7 11 13 17 19 23 29 31 37 41
```

Clasa a VIII-a

P020407: 7 cifre

Se consideră un număr format din șapte cifre `abcdefg`. Se cere să se determine fracția `(abc + fg) / (a + b + c + d + e + f + g)`, precum și fracția simplificată corespunzătoare. Prin `abc` a fost notat numărul format din primele trei cifre ale numărului dat, iar prin `fg` a fost notat numărul format din ultimele două cifre. Numitorul fracției este obținut prin însumarea cifrelor numărului dat.

Date de intrare

De la *intrarea standard* (tastatură) se va citi de pe prima linie un singur număr natural format din șapte cifre.

Date de ieșire

La *ieșirea standard* (ecran), pe prima linie se va afișa numărătorul fracției cerute în enunț, caracterul "/" și numitorul fracției.

Pe cea de-a doua linie se va afișa numărătorul fracției simplificate, caracterul "/" și numitorul fracției simplificate.

Exemplu

Intrare standard

```
4219650
```

Ieșire standard

```
471/27
157/9
```

P020408: Putere

Se consideră un vector care conține *n* elemente și un număr natural $k > 1$.

Elementele vectorului sunt numere naturale formate din cel mult patru cifre.

Se cere să se determine care este puterea maximă $p > 0$ la care poate fi ridicat numărul *k*, astfel încât produsul elementelor vectorului să se dividă cu k^p.

Date de intrare

De la *intrarea standard* (tastatură) se va citi de pe prima linie un singur număr natural *n* care reprezintă numărul de componente care se află în vector.

De pe cea de-a doua linie se vor citi *n* numere reale, care reprezintă valorile componentelor vectorului.

De pe cea de-a treia linie se va citi un singur număr natural care reprezintă valoarea lui *k*.

Date de ieșire

La *ieșirea standard* (ecran) se va afișa valoarea puterii *p* determinată astfel încât să se respecte condiția din enunț (produsul elementelor vectorului să se dividă cu k^p).

Exemplu

Intrare standard

```
6
2 17 32 41 58 18
2
```

Ieșire standard

```
8
```

Clasa a IX-a

P020409: Descompunere

Să se determine toate posibilitățile de a scrie un număr natural N dat ca sumă de numere naturale consecutive.

Date de intrare

Din fișierul de intrare **DATE.IN** se citește numărul N.

Date de ieșire

În fișierul de ieșire **DATE.OUT** se vor scrie pe câte un rând numerele naturale consecutive ale căror sumă este N. Numerele dintr-o descompunere vor fi separate prin câte un spațiu.

Restricții și precizări

- $3 \le N \le 1000$;
- datele de intrare sunt alese întotdeauna astfel încât să existe cel puțin o soluție.

Exemplu

DATE.IN

```
55
```

DATE.OUT

```
1 2 3 4 5 6 7 8 9 10
9 10 11 12 13
27 28
```

Timp maxim de execuție/test: 1 secundă

P020410: Transformare

Dându-se N numere naturale, trebuie să se stabilească numărul total al cifrelor egale cu 1 în cele N numere și dacă numărul cifrelor egale cu 1 este diferit de 0, să se păstreze acest număr.

După efectuarea acestei operații, trebuie să se stabilească numărul total al cifrelor egale cu 2 în cele N numere date și dacă acest număr este diferit de 0, să se păstreze și acesta.

Se va continua numărătoarea pentru toate celelalte cifre, cu excepția cifrei 0.

În continuare, numerele obținute vor fi prelucrate, în mod repetat, conform procedeului aplicat celor N numere date inițial, *până în momentul în care numărul obținut are o singură cifră.*

Date de intrare

De pe prima linie a fișierului de intrare **SIR.IN** se citește numărul N.

Pe linia următoare a fișierului se află N numere naturale, separate prin câte un spațiu.

Date de ieșire

Pe fiecare linie a fișierului de ieșire **SIR.OUT** se vor scrie numerele obținute la o prelucrare.

Ultima linie trebuie să conțină un singur număr format dintr-o singură cifră. Numerele se vor despărți prin câte un spațiu.

Restricții și precizări

- $3 \le N \le 100$;
- cele N numere sunt cuprinse între 0 și 1.000.000.000;
- cifrele egale cu 0 nu se numără;
- atunci când o cifră nu apare niciodată, în șirul care se construiește nu se adaugă cifra 0.

Exemplu

SIR.IN

```
4
123111 2071 111 1411817
```

SIR.OUT

```
12 2 1 1 2 1
4 3
1 1
2
```

Explicație

La prima prelucrare obținem douăsprezece cifre '1', două cifre de '2', o cifră '3', o cifră '4', nici o cifră '5' (nu se păstrează), nici o cifră '6' (nu se păstrează), două cifre '7', o cifră '8' și nici o cifră '9' (nu se păstrează).

La a doua prelucrare avem patru cifre '1', trei cifre '2' și nici o altă cifră.

La a treia prelucrare avem o cifră '3', o cifră '4' și nici o altă cifră.

La a patra prelucrare avem doar două cifre '1', deci prelucrările se opresc.

Timp maxim de execuție/test: 1 secundă

P020411: Bomboane

Mașina de ambalat bomboane de ciocolată a lui *Moș Crăciun* învelește un număr de 30 de bomboane (sosite pe banda rulantă în șir) în staniol alb, galben și roșu.

Mașina pune aceste bomboane în pachete respectând regulile:

- prima bomboană va fi pusă într-un pachet nou (în primul);
- de fiecare dată când o bomboană este învelită în staniol de aceeași culoare cu cel al bomboanei precedente, va închide pachetul curent și va pune bomboana respectivă într-un nou pachet (în următorul);
- ultima bomboană din cele 30 va fi ultima din ultimul pachet.

Date de intrare

Fișierul de intrare **BOMBOANE.IN** conține o singură linie pe care se află 30 de valori numere întregi separate prin câte un spațiu, reprezentând culoarea staniolului în care au fost învelite cele 30 de bomboane; numerele pot avea valoarea 1 (pentru staniol alb), 2 (pentru staniol galben) sau 3 (pentru staniol roșu).

Date de ieșire

Singura linie din fișierul de ieșire **BOMBOANE.OUT** va conține cel mai mare număr de bomboane care se află într-un pachet în care se află cel puțin o bomboană care este învelită în staniol roșu.

Restricție

- în șirul de 30 de bomboane cel puțin o bomboană este învelită în staniol de culoare roșie.

Exemplu

BOMBOANE.IN

```
1 2 1 3 2 3 1 1 2 2 3 1 2 3 1 1 3 1 2 1 1 2 1 2 1 2 1 2 2 3
```

BOMBOANE.OUT

```
7
```

Explicație

Pachetele obținute sunt scrise în ordine pe următoarele linii. Aceste informații nu trebuie scrise în fișierul de ieșire!

Pachet 1: 1 2 1 3 2 3 1
Pachet 2: 1 2
Pachet 3: 2 3 1 2 3 1
Pachet 4: 1 3 1 2 1
Pachet 5: 1 2 1 2 1 2 1 2
Pachet 6: 2 3

Se observă, că printre pachete, cel mai "mare" este pachetul cu numărul 5, dar acesta nu conține nici o bomboană învelită în staniol de culoare roșie.

Ca urmare, cel mai mare pachet care conține cel puțin o bomboană învelită în staniol roșu este primul pachet, format din șapte bomboane.

Timp maxim de execuție/test: 1 secundă

Clasa a X-a

P020412: Dreptunghiuri

N dreptunghiuri de diferite culori se suprapun succesiv pe o foaie de hârtie dreptunghiulară de dimensiuni $a \times b$, colorată în culoarea c.

Dreptunghiurile se așează paralel cu laturile hârtiei și cad în interiorul acesteia. Privind de sus, se obțin diferite figuri colorate.

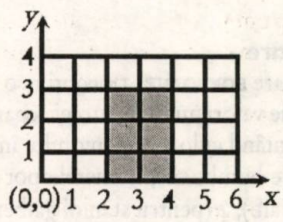


Figura 1

O *figură* este formată din regiuni vecine de aceeași culoare. Două regiuni sunt vecine dacă au cel puțin un punct comun.

Să se determine aria maximă a unei figuri formate pe hârtie și culoarea sa.

Sistemul de coordonate xOy are originea în colțul din stânga-jos al foii, iar axele acestuia sunt paralele cu laturile hârtiei.

Pentru coordonatele 2 3 și 4 0, dreptunghiul va arăta ca în figura 1.

Date de intrare

Datele de intrare se citesc din fișierul **DREPT.IN** astfel:

- prima linie a fișierului conține patru numere naturale a, b, c și n, separate între ele prin câte un spațiu, reprezentând dimensiunile și culoarea foii, respectiv numărul dreptunghiurilor;
- următoarele n linii conțin fiecare câte cinci numere naturale, separate prin câte un spațiu, reprezentând:
  - coordonatele colțului din stânga-sus (x_1, y_1) al dreptunghiului;
  - coordonatele colțului din dreapta-jos (x_2, y_2);
  - culoarea dreptunghiului.

Ordinea de așezare a celor n dreptunghiuri pe foaie este dată de ordinea liniilor în care se află descrierea lor în fișierul de intrare și, în momentul amplasării unui dreptunghi, culoarea acestuia acoperă culoarea existentă sub dreptunghi.

Date de ieșire

Aria și culoarea figurii de suprafață maximă se vor afișa pe prima linie a fișierului **DREPT.OUT**, separate printr-un spațiu.

Restricții și precizări

- valorile a, b și c sunt numere naturale;
- culorile și coordonatele colțurilor dreptunghiurilor sunt codificate cu numere naturale;
- aria se exprimă prin numărul pătrățelelor de latură unitate, aparținând figurii având aceeași culoare;
- $0 < N < 200$;
- $0 < a, b < 30$;
- $0 < c < 70$;
- datele de intrare sunt corecte.

Exemplu

DREPT.IN

```
10 8 2 8
0 7 4 0 1
1 8 8 5 5
2 6 9 3 3
1 2 9 0 5
6 3 8 0 1
5 8 7 5 2
9 8 10 6 3
7 4 10 1 1
```

DREPT.OUT

```
19 3
```

Timp maxim de execuție/test: 2 secunde

P020413: Formula magică

Harry Potter, celebrul personaj al cărților lui *J. K. Rowling*, a ajuns din nou într-o încurcătură. Este pedepsit și trebuie să stea închis într-o cameră. Ca să poată ieși, este nevoit să găsească o formulă magică în *Manualul Standard de Vrăji*.

Ajutați-l pe *Harry Potter* să poată ieși din camera de pedeapsă. Pentru aceasta trebuie să răsfoiți *Manualul Standard de Vrăji* și să găsiți cea mai lungă înșiruire de caractere consecutive din manual, secvență care citită de la cap la coadă și citită de la coadă la cap este aceeași.

Date de intrare

În fișierul de intrare **VRAJI.IN** se află un text, scris pe mai multe linii, care conține *Manualul Standard de Vrăji*. Se vor folosi doar caractere care se găsesc în setul *ASCII* de bază.

Date de ieșire

Pe prima linie a fișierului de ieșire **VRAJI.OUT** se va scrie un număr care reprezintă lungimea șirului de caractere găsit, iar pe următoarea linie se va scrie una dintre secvențele de caractere de lungime maximă.

Restricții și precizări

- șirul de caractere care constituie formula magică este scris pe o singură linie;
- din formula magică pot face parte orice caractere, mai puțin marca de sfârșit de linie (<Enter>);
- majusculele și literele mici se consideră a fi egale;
- dacă în manual nu se găsește nici o formulă magică, atunci fișierul de ieșire va conține o singură linie pe care se va afla mesajul "NU";
- o formulă magică trebuie să conțină cel puțin două caractere;
- fișierul de intrare conține cel mult 1000 linii, iar o linie cel mult 255 caractere;
- datele de intrare sunt astfel alese încât să existe soluție unică.

Exemple

VRAJI.IN

```
Oglinda ne arata cea mai ascunsa,
cea mai arzatoare dorinta a
noastra. Ministerul Magiei a
decis:Un nu hotarat Vrajilor de
Transfigurare.
```

VRAJI.OUT

```
5
Un nu
```

VRAJI.IN

```
Harry, a sosit timpul sa iti
marturisesc ceea ce ar fi trebuit
sa-ti spun cu cinci ani in urma.
Te rog sa inchizi aerisirea si sa
iei loc. Iti voi spune totul...
```

VRAJI.OUT

```
11
 aerisirea
```

Explicație

În cazul celui de-al doilea exemplu, secvența corectă considerată conține și spațiul dinaintea cuvântului, respectiv cel de după cuvânt.

Timp maxim de execuție/test: 1 secundă

P020414: Relații de familie

Se cunosc persoanele de sex feminin din mai multe familii. Informațiile despre ele sunt precizate sub forma unor relații de tipul *MAMĂ - FIICĂ*.

Să se determine:

- cel mai mare număr de descendenți (urmași) pe care îi are o persoană: fiice, nepoate, etc;
- persoana care are numărul maxim de descendenți.

Date de intrare

Pe prima linie a fișierului de intrare **FAMILII.IN** este scris un număr natural n care reprezintă numărul de relații *MAMĂ - FIICĂ*.

Pe următoarele n linii se află perechi de două nume: numele mamei urmat de numele fiicei, separate printr-un singur spațiu.

Date de ieșire

Fișierul de ieșire **FAMILII.OUT** trebuie să conțină mai multe linii:

- pe prima linie se va scrie numărul maxim al descendenților unei persoane;
- pe următoarele linii se vor scrie numele tuturor persoanelor care au număr maxim de descendenți, câte un nume pe o linie.

Restricții și precizări

- $1 \le n \le 100$;
- nu există două persoane cu același nume;
- dacă există mai multe persoane care au numărul maxim de descendenți, se vor scrie în fișierul de ieșire numele tuturor acestor persoane;
- datele de intrare sunt corecte;
- numele sunt șiruri de caractere formate din litere mari ale alfabetului englez; cele două nume corespunzătoare unei relații sunt șiruri de caractere ale căror lungime totală este cel mult 255.

Exemplu

FAMILII.IN

```
7
ANA DANA
GETA CRINA
MARIA FLAVIA
GETA MARIA
DORA ELENA
CRINA ANA
DANA CRISTINA
```

FAMILII.OUT

```
6
GETA
```

Timp maxim de execuție/test: 1 secundă

Clasele a XI-a și a XII-a

P020415: Frății

Locuitorii unei galaxii îndepărtate au descoperit că pot călători rapid între sistemele planetare ale galaxiei folosind porți stelare care îi pot conduce rapid de la un sistem la alt sistem.

Deoarece toate aceste porți funcționează doar într-un singur sens, locuitorii galaxiei au hotărât ca toate sistemele între care se poate călători în ambele sensuri (direct sau prin intermediul altor sisteme) să formeze o frăție.

Cunoscând sistemele planetare și punctul de plecare, respectiv destinația fiecărei porți din galaxie, determinați numărul frățiilor din galaxia respectivă.

Date de intrare

Prima linie a fișierului de intrare **FRATII.IN** conține o valoare N, reprezentând numărul sistemelor planetare.

Pe a doua linie se află o valoare M, reprezentând numărul porților stelare.

Pe fiecare dintre următoarele M linii sunt scrise două numere (x, y) separate printr-un spațiu, x reprezentând numărul de ordine al sistemului de plecare, iar y numărul de ordine al sistemului care constituie destinația unei porți stelare.

Date de ieșire

Fișierul de ieșire **FRATII.OUT** trebuie să conțină o singură linie pe care se va afla un număr natural care reprezintă numărul frățiilor din galaxie.

Restricții și precizări

- N este un număr întreg și are valoarea cuprinsă între 0 și 100 inclusiv;
- sistemele planetare sunt identificate prin numere întregi de la 1 la N;
- $1 \le x, y \le N$;
- M este un număr întreg și are valoarea cuprinsă între 0 și 9900 inclusiv.

Exemple

FRATII.IN

```
2
1
2 1
```

FRATII.OUT

```
2
```

FRATII.IN

```
3
4
1 2
2 3
3 1
1 3
```

FRATII.OUT

```
1
```

FRATII.IN

```
4
4
1 3
3 1
4 2
4 3
```

FRATII.OUT

```
3
```

Explicație

În cazul primului exemplu, fiecare sistem formează câte o frăție, deoarece între cele două sisteme nu se poate călători în ambele sensuri.

În cazul celui de-al doilea exemplu, există o singură frăție, deoarece între sistemele 1 și 3 se poate călători direct în ambele sensuri, între 1 și 2 direct într-un sens și prin intermediul sistemului 3 în celălalt sens, iar între 2 și 3 se procedează similar.

În cazul celui de-al treilea exemplu, există o frăție formată din sistemele 1 și 3, una formată din sistemul 2 și o alta din sistemul 4, deci trei frății.

Timp maxim de execuție/test: 1 secundă

P020416: Bloc de marmură

Dintr-o carieră s-a extras un bloc de marmură sub forma unui paralelipiped dreptunghic. În urma analizelor microstructurale s-au constatat deficiențe de structură în diverse puncte ale blocului.

Știind că, din motive tehnice, blocul nu poate fi tăiat decât paralel cu oricare dintre fețele sale, determinați volumul celui mai mare bloc (paralelipiped dreptunghic) care se poate obține din blocul inițial și care să nu aibă deficiențe de structură.

Date de intrare

Pe prima linie a fișierului de intrare **BLOC.IN** sunt scrise dimensiunile *a*, *b* și *c* ale blocului inițial (separate prin câte un spațiu).

Pe a doua linie se află numărul *N* al deficiențelor de structură, iar pe următoarele *N* linii, triplete *x*, *y*, *z*, reprezentând coordonatele deficiențelor (separate prin spațiu).

Date de ieșire

Fișierul de ieșire **BLOC.OUT** trebuie să conțină o singură linie pe care se va afla valoarea *V* care reprezintă volumul celui mai mare bloc care nu are deficiențe de structură.

Restricții și precizări

- blocul de marmură inițial are un colț în originea sistemului de axe *xOyz* și laturile paralele cu axele de coordonate;
- dimensiunile *a*, *b* și *c* corespund lungimilor laturilor pe axele de coordonate *Ox*, *Oy*, respectiv *Oz*;
- deficiențele sunt punctiforme, coordonatele lor *x*, *y* și *z* corespund unor distanțe pe axele de coordonate *Ox*, *Oy* și respectiv *Oz* față de originea sistemului de coordonate;
- *a*, *b*, *c*, *N* și valorile tripletelor *x*, *y*, *z* sunt numere întregi;
- $0 \le N \le 30$;
- $1 \le x < a \le 1000$;
- $1 \le y < b \le 1000$;
- $1 \le z < c \le 1000$.

Exemplu

BLOC.IN

```
500 400 600
3
100 100 100
300 200 500
400 300 300
```

BLOC.OUT

```
64000000
```

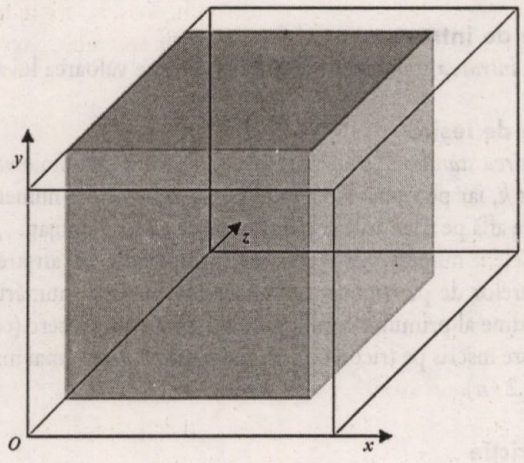


Figura 2

Explicație

Coordonatele blocului obținut vor fi $x_1 = 0$, $x_2 = 400$, $y_1 = 0$, $y_2 = 400$, $z_1 = 100$, $z_2 = 500$, reprezentat schematic în figura 2.

Timp maxim de execuție/test: 1 secundă

P020417: Număr mic

Se consideră un număr natural *N*. Determinați cel mai mic număr natural *M* care se poate obține din numărul *N*, prin eliminarea unui număr de *K* cifre din acesta, fără a modifica ordinea cifrelor rămase.

Date de intrare

Fișierul de intrare **NUMARMIC.IN** conține o singură linie pe care se află valorile *N* și *K*, separate printr-un spațiu.

Date de ieșire

Fișierul de ieșire **NUMARMIC.OUT** trebuie să conțină o singură linie pe care se va afla valoarea *M*.

Restricții și precizări

- $10 \le N \le 10^{15}$;
- $1 \le K <$ numărul de cifre al numărului *N*;
- rezultatul trebuie să folosească toate cifrele rămase, nefiind permise zerouri nesemnificative.

Exemple

NUMARMIC.IN

```
3195842 3
```

NUMARMIC.OUT

```
1542
```

NUMARMIC.IN

```
10000100000 8
```

NUMARMIC.OUT

```
100
```

NUMARMIC.IN

```
2047 3
```

NUMARMIC.OUT

```
0
```

Timp maxim de execuție/test: 1 secundă

Notă

Enunțurile mai multor probleme au fost modificate de către redacția *GInfo* pentru îmbunătățirea clarității.

ianuarie 2004, Târgu-Jiu

Olimpiada LOCALĂ de Informatică

În cursul lunii ianuarie a avut loc la Colegiul Național "Ecaterina Teodoroiu" din Târgu-Jiu ediția din acest an a fazei locale a olimpiadei de informatică. La acest concurs au participat elevi din clasele IX - XII care s-au confruntat cu câte două probleme. În continuare vă prezentăm enunțurile celor șase probleme propuse spre rezolvare.

Clasa a IX-a

P020418: Șir

Se consideră un număr natural n și următorul șir de numere întregi:

1, -1, 1, 1, 2, 2, -1, -1, -2, -2, 1, 1, 1, 2, 2, 2, 3, 3, 3, -1, -1, -1, -2, -2, -2 , -3, -3, -3, ...

Se cere să se determine cel de-al n-lea element al acestui șir.

Date de intrare

De la *intrarea standard* (tastatură) se citește un singur număr natural care reprezintă valoarea lui n.

Date de ieșire

La *ieșirea standard* (pe ecran) se va scrie cel de-al n-lea termen al șirului din enunț

Restricție

- $1 \le n \le 500.000.000$.

Exemplu

Intrarea standard

```
2
```

Ieșirea standard

```
-1
```

Intrarea standard

```
10
```

Ieșirea standard

```
-2
```

Intrarea standard

```
1000
```

Ieșirea standard

```
-10
```

P020419: Copii

Considerăm că $2 \cdot n$ copii au tricouri numerotate cu numere de la 1 la $2 \cdot n$.

Sarcina voastră este să așezați copiii în cerc astfel încât, eliminându-i din k în k, mai întâi să părăsească cercul, în ordine descrescătoare a numerelor, cei cu numere prime înscrise pe tricou, apoi, în ordine crescătoare a numerelor, cei cu numere neprime pe tricou.

Se cunoaște valoarea lui n, iar valoare lui k este dată de cel mai mare număr mai mic decât n care este un pătrat perfect.

Date de intrare

De la *intrarea standard* (tastatură) se citește valoarea lui n.

Date de ieșire

La *ieșirea standard* (pe ecran) se va scrie pe prima linie valoarea k, iar pe următoarea linie se vor scrie cele n numere care se află pe tricourile copiilor după ce au fost aranjați.

Cele n numere vor fi separate prin spațiu, iar afișarea numerelor de pe tricouri se va face începând cu numărul de ordine al primului copil care va fi eliminat din cerc (cel care are înscris pe tricou cel mai mare număr prim, mai mic decât $2 \cdot n$).

Restricție

- $2 \le n \le 500$.

Exemplu

Intrarea standard

```
8
```

Ieșirea standard

```
4
13 3 4 12 11 10 2 14 7 6 9 1 5 16 15 8
```

Clasa a X-a

P020420: Expresie

Se consideră o expresie formată folosind numere naturale și perechi de paranteze drepte.

Să se calculeze valoarea acestei expresii, știind că includerea între paranteze corespunde operației de calcul a câtului împărțirii întregi la 2, iar alăturarea a două paranteze corespunde operației de adunare a valorilor celor două subexpresii.

Expresia poate fi calculată doar dacă este corectă, adică nu conține numere care să nu fie incluse între paranteze drepte, nu conține perechi de paranteze care să nu includă nici un număr sau nici o altă subexpresie și perechile de paranteze se închid corect din punct de vedere sintactic.

Date de intrare

Pe prima linie a fișierului de intrare **EXPRESIE.IN** se află o expresie formată numai din paranteze drepte și cifre de la 0 la 9.

Date de ieșire

În fișierul de ieșire **EXPRESIE.OUT** se va scrie valoarea expresiei în cazul în care aceasta este corectă din punct de vedere sintactic.

Dacă expresia din fișierul de intrare nu poate fi calculată datorită faptului că nu este corectă din punct de vedere sintactic, atunci pe prima linie a fișierului de ieșire se va scrie mesajul `"expresie gresita"`.

Restricții și precizări

- numerele care apar în expresii sunt întregi și sunt cuprinse între 0 și 30.000;
- expresia din fișierul de intrare este formată din cel mult 200 de caractere;
- datele de intrare sunt corecte.

Exemplu

EXPRESIE.IN

```
[[34][23[12][]
```

EXPRESIE.OUT

```
expresie gresita
```

EXPRESIE.IN

```
[4][[6][10]]
```

EXPRESIE.OUT

```
6
```

Explicație

În cazul primului exemplu expresia este greșită din punct de vedere sintactic deoarece parantezele nu se închid cum trebuie (de exemplu, prima paranteză nu se închide), nu toate numerele sunt incluse între paranteze și nu toate perechile de parateze `'['` și `']'` conțin elemente între ele.

În cazul celui de-al doilea exemplu valoarea expresiei este 6 deoarece `[6] = 3`, `[10] = 5`, `[4] = 2`, `[[6][10]] = [(3+5)] = 4`, deci `[4][[6][10]] = 2 + 4 = 6`.

P020421: Șir 2

Profesoara voastră de matematică este "cam exigentă" și vă dă foarte multe calcule de făcut, ca temă pentru acasă. Voi nu vă lăsați descurajați și folosiți calculatorul pentru efectuarea calculelor. Însă profesoara s-a prins de acest lucru și, ca să nu mai folosiți calculatorul, vă propune următoarea problemă:

"*Se consideră următorul șir, construit astfel încât fiecare element al lui, cu excepția primului, se obține din cel precedent*: `1, 11, 21, 1211, 111221, ...`

Termenii din șir sunt numerotați începând cu numărul de ordine 1.

Sarcina voastră este să determinați, pe baza numărului natural n, cel de-al n-lea termen din acest șir.

Profesoara a propus problema respectivă la prima oră a dimineții, urmând ca după o oră să dați răspunsul. Între timp, aveți oră de informatică și, pentru a vă verifica rezultatul găsit, v-ați hotărât să realizați un program care să calculeze valoarea celui de-al n-lea termen din șirul propus de doamna profesoară.

Date de intrare

În fișierul de intrare **SIR.IN** se află numărul natural n, care reprezintă indicele elementului pe care va trebui să-l determinați, din șirul prezentat în enunț.

Date de ieșire

În fișierul de ieșire **SIR.OUT** trebuie să se afle linie valoarea celui de-al n-lea element al șirului.

Restricție

- $3 \leq n \leq 50$.

Exemple

SIR.IN

```
4
```

SIR.OUT

```
1211.
```

SIR.IN

```
5
```

SIR.OUT

```
111221
```

Clasele a XI-a și a XII-a

P020422: Sateliți

Se preconizează lansarea pe orbită a unui grup de sateliți care pot schimba date între ei.

Doi sateliți pot transmite date între ei în mod direct sau în mod indirect (adică, prin intermediul altor sateliți); de asemenea, este posibil să existe sateliți care să nu aibă nici o legătură cu alții. Spunem că doi sateliți sunt vecini numai dacă ei pot transmite unul altuia date în mod direct.

Fiind date un număr n și un set de n numere întregi pozitive, se cere :

- să se stabilească dacă există o configurație de n sateliți astfel încât setul de n numere citite la intrare să corespundă numerelor de vecini ai sateliților;
- dacă răspunsul la punctul anterior este "DA", atunci descrieți o astfel de configurație;
- fiind dată configurația de la punctul anterior, stabiliți dacă se pot trimite date între oricare doi sateliți.

Date de intrare

În fișierul de intrare **SATELITI.IN** sunt scrise pe o singură linie numere naturale, separate între ele prin spațiu.

Primul număr de pe această linie reprezintă valoarea numărului de sateliți n care au fost lansați pe orbită. În continuare se află un șir de n numere naturale care reprezintă numărul de vecini pe care trebuie să-i aibă un satelit din mulțime.

Date de ieșire

Prima linie a fișierului de ieșire **SATELITI.OUT** va conține un mesaj care constituie răspunsul la primul punct și poate fi DA sau NU.

Dacă răspunsul este NU, nu se mai cere altceva la ieșire.

Dacă răspunsul este DA, atunci fiecare dintre următoarele n linii va conține n numere, separate prin spații, care pot avea valoarea 0 sau 1. Numerele de pe aceste n linii formează o matrice pătratică a de dimensiune n.

Doi sateliți $1 \le i \le n$ și $1 \le j \le n$ sunt vecini dacă și numai dacă $a_{i,j} = 1$, iar $a_{i,i} = 0$, $\forall\ 1 \le i \le n$.

A $(n+2)$-a linie a fișierului de ieșire corespunde răspunsului pentru ultimul punct al cerinței problemei și va conține unul dintre mesajele DA sau NU.

Restricții și precizări

- $1 \le n \le 20$;
- ordinea în care sunt date numerele vecinilor direcți nu trebuie respectată;
- datele din fișierul de intrare sunt corecte.

Exemple

SATELITI.IN
```
6 4 3 1 4 2 0
```

SATELITI.OUT
```
NU
```

SATELITI.IN
```
7 4 3 1 5 4 2 1
```

SATELITI.OUT
```
DA
0 1 1 1 1 1 0
1 0 1 0 0 0 0
1 1 0 1 0 1 0
1 0 1 0 0 1 0
1 0 0 0 0 0 0
1 0 1 1 0 0 1
0 0 0 0 0 1 0
DA
```

P020423: Funcție

Se consideră un număr natural n și o funcție:

$f: A \to A$, unde $A = \{1, 2, \ldots, n\}$.

Se cere să se detemine o submulțime S a lui A care să aibă un număr maxim de elemente astfel încât restricția lui f la S să fie o funcție injectivă și $f(S) = S$.

Date de intrare

Fișierul de intrare **FUNCTIE.IN** conține pe prima linie un număr natural n, care reprezintă numărul de elemente ale mulțimii A. Pe a doua linie se află n numere naturale, separate prin spațiu, care reprezintă valorile funcției f pentru fiecare element din mulțimea A.

Date de ieșire

Fișierul de ieșire **FUNCTIE.OUT** va conține caracterul '{', elementele submulțimii determinate S separate între ele prin caracterul ',' și apoi caracterul '}', după cum se poate observa și în exemplul următor.

În fișierul de ieșire nu trebuie să existe spații.

Restricții și precizări

- $1 \le n \le 1000$.

Exemplu

FUNCTIE.IN
```
7
3 1 1 5 5 4 6
```

FUNCTIE.OUT
```
{1,3,5}
```

FUNCTIE.IN
```
6
1 2 3 4 5 6
```

FUNCTIE.OUT
```
{1,2,3,4,5,6}
```

Kenosha, Statele Unite ale Americii

IOI 2003

În perioada 16-23 august 2003 a avut loc cea de-a XV-a ediție a concursului internațional de programare IOI. În continuare vă vom prezenta soluțiile problemelor propuse spre rezolvare la acest concurs.

P060301: Comparări de cod

La început, vom înlocui, în ambele programe, numele variabilelor prin indici. În acest mod, detaliile legate de o variabilă vor fi accesate în $O(1)$.

Presupunem alese două secvențe de L linii, câte una din fiecare program. Să vedem cum putem verifica dacă aceste secvențe se potrivesc.

Fie linia i din fiecare set. Este clar că variabilele din stânga semnului "=" trebuie să se potrivească. Marcăm această potrivire. Numărul de operații este constant pentru fiecare linie.

Dacă în partea dreaptă se află aceeași variabilă de două ori, este clar că ea trebuie să se potrivească. Avem, din nou, număr constant de operații pentru fiecare linie.

Dacă variabila X nu apare decât în partea dreaptă a ecuațiilor din unul din programe, atunci toate ecuațiile corespunzătoare din programul celălalt trebuie să aibă cel puțin o variabilă în comun. Acest pas se poate rezolva în $O(L)$.

Dacă am stabilit că o variabilă X se potrivește cu o variabilă Y (din celălalt program) și ulterior obținem că se potrivește cu o variabilă Z, atunci secvențele de linii nu se potrivesc.

Condițiile se verifică pentru ambele programe. Nerespectarea unei condiții înseamnă că secvențele de linii nu se potrivesc. Este evident că aceste condiții sunt necesare pentru ca secvențele să se potrivească. Se poate demonstra că ele sunt și suficiente.

Partea mai puțin intuitivă este cea în care nu știm dacă o variabilă X din partea dreaptă a unei linii se potrivește cu Y sau cu Z, și verificăm dacă toate liniile corespondente liniilor în care apare X au cel puțin o variabilă în comun.

Să analizăm cazurile care apar:

- liniile nu au nici o variabilă în comun; atunci, secvențele de L linii nu se potrivesc;
- liniile au două variabile în comun; atunci, în toate liniile în care apare X apare o altă variabilă T, și putem potrivi fie X cu Y și T cu Z, fie invers. În caz contrar, secvențele nu se potrivesc.
- liniile au o singură variabilă în comun; atunci, ea corespunde lui X.

Folosind această metodă de verificare putem construi o primă variantă de rezolvare: alegem, pe rând, câte o linie din fiecare program și determinăm lungimea celei mai lungi secvențe care se potrivește, începând cu liniile respective. Determinarea se realizează cu ajutorul metodei de mai sus, folosind căutarea binară pentru alegerea lungimii.

Acest algoritm ar fi obținut aproximativ 55% din punctajul total.

Pentru a obține punctajul maxim se procedează în felul următor:

- se alege, pe rând, o diferență între indicii liniilor la care se caută cele două secvențe identice. Diferența variază de între $-R$ la R. Astfel, dacă o secvență începe la linia a treia a programului *RBN*, iar cealaltă la linia 7 din programul *HAL*, diferența este 4.
- pentru fiecare diferență aleasă se pornește de la primele două linii cu diferența respectivă între numerele lor de ordine (de exemplu, de la liniile 1 și 5 pentru diferența 4) și se construiește o secvență cât mai lungă care să se potrivească, începând cu aceste două linii. Aceasta se face prin adăugarea de linii la sfârșitul secvențelor, cât timp condițiile sunt respectate. În momentul în care condițiile sunt încălcate, se elimină linii de la începutul secvențelor, până când condițiile sunt din nou respectate.

Acest algoritm necesită actualizări incrementale legate de respectarea condițiilor, la adăugarea și eliminarea unei linii. Pentru ca algoritmul să fie eficient, actualizările trebuie să se realizeze în timp constant.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(R + H)$, iar cel al operației de scriere a rezultatelor este $O(1)$.

Pentru fiecare diferență cuprinsă între $-R$ și R se efectuează $O(H)$ operații, presupunând că actualizarea se face în

timp constant, deci ordinul de complexitate al acestei operații este $O(R \cdot H)$.

În final, ordinul de complexitate al algoritmului de rezolvare pentru această problemă este $O(R \cdot H)$.

P060302: Întreținerea potecilor

Această problemă admite două variante eficiente de rezolvare. Ambele rezolvări ar fi obținut punctajul maxim în cadrul acestui concurs.

Prima variantă constă în a reconstrui, după fiecare săptămână, arborele parțial de cost minim, folosind algoritmul lui *Kruskal*.

Arborele parțial determinat după o anumită săptămână poate conține numai muchii din arborele parțial anterior, plus muchia introdusă în săptămâna respectivă.

Acest fapt este evident dacă ne gândim la modul de funcționare a algoritmului lui *Kruskal*.

Muchiile sunt sortate după cost; muchia nouă ar intra pe o anumită poziție.

Presupunem că reconstruim arborele. Este clar că, parcurgând lista de muchii până înainte de poziția muchiei nou introduse, se obține aceeași configurație ca la construirea precedentă a arborelui.

Dacă muchia nouă ar fi respinsă, deoarece ar introduce un ciclu, s-ar obține același arbore. Dacă muchia nouă unește două componente, atunci una dintre muchiile care au fost în arborele precedent va fi respinsă la un moment dat ca introducând un ciclu și, ceea ce este mai important, nici o muchie care a fost respinsă la construirea arborelui de la pasul precedent nu ar fi acceptată, deoarece capetele ei ar fi în aceeași componentă.

Se poate face o demonstrație formală prin inducție.

A doua variantă constă în construirea arborelui de la un anumit pas pe baza arborelui de la pasul precedent.

La introducerea unei noi muchii, între capetele ei există un singur drum în arbore. Dacă muchia de lungime maximă de pe acest drum este mai lungă decât muchia nouă, atunci este de preferat să o eliminăm și să o introducem pe cea nouă. Astfel se obține un arbore cu un cost mai bun.

Extinderea acestor doi algoritmi pentru a funcționa și la început, când încă nu s-a format un arbore, este foarte simplă. Nici una din variante nu ridică dificultăți deosebite la implementare.

Analiza complexității

Operațiile de citire a datelor de intrare și scriere a rezultatelor au ordinul de complexitate $O(W)$.

Primul algoritm constă în determinarea a W arbori parțiali, pentru N grafuri având cel mult N muchii fiecare. Deoarece algoritmul lui *Kruskal* pentru determinarea arborelui parțial de cost minim al unui graf are ordinul de complexitate $O(M \cdot \log N)$, unde M este numărul total de muchii al grafului, ordinul de complexitate al acestui algoritm este $O(W \cdot N \cdot \log N)$.

În cazul celui de-al doilea algoritm, adăugarea unei noi muchii se face în $O(N)$, deci acesta are ordinul de complexitate $O(W \cdot N)$.

Se poate observa că ordinul de complexitate al rezolvării acestei probleme nu depinde de ordinul de complexitate al operațiilor de citire a datelor de intrare și scriere a rezultatelor, ci depinde numai de ordinul de complexitate al algoritmului utilizat.

P060303: Inversare

Pentru această problemă, concurenții cunoșteau datele de intrare; sarcina lor era să furnizeze fișierele de ieșire corespunzătoare.

Pentru $N < 9$, se inițializau regiștrii cu valorile N, N-1, ..., 0 și se afișau aceste valori. Se efectuau 0 operații S.

Pentru valori mai mari ale lui N se folosesc metode euristice. Să examinăm una dintre acestea.

Evident, unul din regiștri va fi blocat pe valoarea 0; în caz contrar, 0 nu poate fi afișat. Astfel, mai rămân 9 regiștri de lucru.

Încercăm să rezolvăm problema folosind o singură operatie S. Inițializăm primul registru cu N, următorul cu N - 2. Afișăm N, apoi primul registru poate fi modificat, deci îl facem egal cu N - 1 printr-o operație S. Afișăm N - 1, apoi putem face un nou S, afișăm N - 2, facem un nou S, și trebuie să afișăm N - 3.

Putem pune N - 5 în al treilea registru. În acest caz, primul S nespecificat ar fi pus primul registru pe N - 4, iar al doilea S ar fi pus al doilea registru pe N - 3, după care s-ar fi afișat N - 3. Al treilea registru rămâne nemodificat până la afișarea lui N - 5, după care poate fi folosit ca registru de lucru, la fel cum au fost folosiți primii doi.

Continuând acest procedeu, se pot rezolva testele pentru care valoarea lui N este mai mică sau egală cu 44.

Generalizând această strategie pentru mai multe operații S consecutive, s-ar fi obținut aproximativ 90 de puncte.

P060304: Frontiera

Considerăm că NR este numărul maxim de vârfuri ale unui poligon.

Considerăm că observatorul (*Fermierul Don*) are un câmp vizual de 360 grade. În acest câmp vizual, la anumite unghiuri, care se vor calcula, se află stâlpii gardului.

Fiecare poligon blochează o porțiune din câmpul vizual al observatorului. Se poate determina această porțiune într-un timp de $O(NR)$.

Porțiunile pot fi puse în corespondență cu intervale închise cu capetele între 0 și 360 de grade, prin "deschiderea" câmpului vizual al observatorului; se observă că poligoanele care intersectează axa de "deschidere" vor conduce la formarea a două intervale.

Se sortează aceste intervale în funcție de punctul de început, apoi, folosind un algoritm similar sortării prin interclasare, se determină stâlpii gardului care nu aparțin nici unui interval.

Analiza complexității

Ordinul de complexitate al operației de citirea a datelor de intrarea este $O(R \cdot NR)$.

Algoritmul de determinare a porțiunilor blocate din câmpul vizual și transformare a lor în intervale are ordinul de complexitate $O(R \cdot NR)$.

Algoritmul de sortare a porțiunilor blocate are ordinul de complexitate $O(R \cdot \log R)$, iar algoritmul de determinare a stâlpilor vizibili are ordinul de complexitate $O(N + R)$.

În concluzie, ordinul de complexiatea al algoritmului de rezolvare pentru această problemă este $O(N + R \cdot (NR + \log R))$.

P060305: Ghicește vaca

În continuare, prin a rezolva o submulțime înțelegem stabilirea unei strategii de a pune întrebări pentru a determina vaca, optimă din punct de vedere al numărului maxim de întrebări, și de calculare a acestui număr.

Abordarea intuitivă constă în a alege la fiecare pas întrebarea care împarte vacile cât mai echitabil, din punct de vedere al numărului de vaci din submulțimile în care se face împărțirea. Această soluție nu este optimă; o împărțire echitabilă nu asigură faptul că submulțimile în care s-a făcut împărțirea se vor rezolva ușor. Astfel, probabil că este preferabil să împărțim vacile în submulțimi, nu neapărat apropiate ca număr de elemente, dar care se pot rezolva mai ușor. În plus, dacă există mai multe întrebări care ar conduce la o împărțire la fel de bună din punct de vedere al numărului de vaci din submulțimi, nu avem nici o modalitate de a alege una dintre ele. În orice caz, o astfel de rezolvare ar fi obținut aproximativ 85 puncte, deci un punctaj mulțumitor.

Rezolvarea optimă a problemei constă în a alege, la fiecare pas, întrebarea care împarte vacile, astfel încât numărul de întrebări pentru rezolvarea celei mai mari submulțimi rezultate din împărțire să fie minim. Această informație nu este cunoscută inițial, deci va trebui să fie calculată.

Rezultă o funcție recursivă care, primind o submulțime a mulțimii vacilor, încearcă, pe rând, fiecare întrebare, se reapelează pentru submulțimile rezultate din împărțire și întoarce numărul minim de întrebări.

Pentru a evita recalcularea unor valori, funcția va testa, la început, dacă valoarea pe care trebuie să o returneze a fost calculată deja, la un apel anterior; această operație se realizează prin memorarea valorilor calculate. Acesta este algoritmul intuitiv; la implementare se vor aduce optimizări de cod.

Aparent există cel mult 2^{50} submulțimi care vor fi testate, deci funcția nu se va încadra în timp. În practică, nu toate cele cel mult 2^{50} submulțimi sunt accesibile folosind întrebările pentru departajare.

Fiecare submulțime care va fi testată poate fi identificată prin mulțimile de valori ale celor cel mult 8 atribute. Pentru un atribut fixat există cel mult șapte mulțimi posibile de valori, excluzând mulțimea vidă ($\{x\}$, $\{y\}$, $\{z\}$, $\{x, y\}$, $\{x, z\}$, $\{y, z\}$). În total există cel mult 7^8 submulțimi, adică mai puțin de 6 milioane de submulțimi.

Implementarea este destul de dificilă, deoarece codul trebuie optimizat.

Este mai eficientă utilizarea unei metode nerecursive similară. Astfel, metoda recursivă realizează o parcurgere în adâncime într-un graf care are drept noduri submulțimile mulțimii vacilor, având câte un arc de la fiecare mulțime la mulțimile în care ea se împarte punând întrebari. Putem înlocui această parcurgere cu o parcurgere în lățime.

Deoarece această metodă se poate obține din cea recursivă cu puține modificări, se pot testa ambele variante și se reține cea mai rapidă.

P060306: Roboții uimitori

Deoarece lungimile traseelor paznicilor sunt mici, se observă că, după 12 minute, toți paznicii se vor întoarce în pozițiile inițiale. Astfel, singura modificare în spațiul problemei, de la minutul i la minutul $i + 12$, constă în pozițiile roboților.

Construim o matrice tridimensională A, în care $A[P_1, P_2, t]$ reprezintă numărul de mutări pentru a aduce primul robot în poziția P_1 și al doilea robot în poziția P_2, la un moment de timp de forma $12 \cdot k + t$. P_1 identifică o poziție a primului robot în primul labirint, sau faptul că acesta a ieșit din labirint. P_2 identifică, în mod similar, starea celui de-al doilea robot.

Elementele matricei A se completează folosind algoritmul lui *Lee*. Practic, o stare este dată de P_1, P_2 și t; din această stare se poate ajunge în cel mult 4 stări, folosind comenzile disponibile. Unele din aceste stări sunt invalide, deoarece duc la capturarea unuia dintre roboți.

Starea inițială este introdusă într-o coadă și la fiecare pas, în coadă sunt introduse apoi stările valide în care se poate ajunge din ea.

Această soluție, dacă ar fi fost implementată corect, ar fi obținut punctajul maxim.

Analiza complexității

Ordinul de complexitate al operației de citirea a datelor de intrare este $O(R_1 \cdot C_1 + G_1 + R_2 \cdot C_2 + G_2)$.

Algoritmul de determinare a comenzilor care trebuie transmise roboților are ordinul de complexitate $O(12 \cdot P_1 \cdot P_2)$, unde P_1 și P_2 reprezintă, în acest caz, numărul de poziții prin care poate trece primul robot, respectiv al doilea robot și aceste numere sunt mărginite superior de $R_1 \cdot C_1$, respectiv $R_2 \cdot C_2$.

Ordinul de complexitate al operației de scriere a rezultatelor este $O(K)$.

Nota redacției

Toate variabilele, care au fost folosite în rezolvarea acestor probleme și nu au fost explicate, au semnificația dată în enunțurile problemelor, care au fost publicate în *GInfo* 13/6 (octombrie 2003).

Münster, Germania

Olimpiada de informatică a EUROPEI CENTRALE

În continuare vă prezentăm soluțiile problemelor propuse spre rezolvare la acest concurs. Aceste soluții au fost realizate de redacția Ginfo pe baza soluțiilor oficiale din limba engleză prezentate de către autorii problemelor.

P060307: Hanoi

Această problemă se reduce la rezolvarea altor două subprobleme. Prima subproblemă constă în a determina tija destinație pentru discuri, iar a doua constă în a determina numărul total de mutări care vor fi efectuate.

Prima subproblemă este relativ simplă, deoarece întotdeauna este mai avantajos să lăsăm discul N (cel mai mare) pe tija pe care acesta se află inițial. Altfel, ar trebui să se mute discul N pe o altă tijă care în momentul respectiv ar trebui să fie liber. Această mutare nu va schimba starea jocului (în afară de renumerotarea tijelor) și astfel nu poate face parte din soluția optimă. În continuare vom nota această tijă destinație prin *dest*.

În continuare vom prezenta modul de rezolvare a celei de-a doua subprobleme care constă în determinarea numărului de mutări.

O abordare algoritmică simplă va fi dedusă din următoarele observații:

- în timpul mutării discului i, pozițiile discurilor N, N - 1, ..., i + 1 (având diametre mai mari) nu au importanță. De exemplu, dacă discurile N, N - 1, ..., i se situează deja pe tija *dest*, nu are sens să se mute nici unul dintre ele încă o dată.
- în scopul de a muta discul i în locul său final pe tija *dest*, toate discurile N, N - 1, ..., i + 1 trebuie să fie deja prezente. Rezultă că discurile trebuie mutate în locul lor final în ordinea descrescătoare a dimensiunilor lor.
- dacă discurile N, N - 1, ..., i + 1 se află deja pe tija *dest*, toate discurile i - 1, ..., 1 trebuie să nu se afle pe aceeași tijă pe care se află discul i, și de asemenea nici pe tija *dest*, datorită regulilor de mutare. Rezultă că aceștia, mai întâi trebuie să fie mutate pe o altă tijă.

Toate aceste restricții conduc la următorul algoritm recursiv:

```
1  Subalgoritm MutăDisc(disc, TijăDest,
                                    TijăDisc):
2    TijăSursă ← TijăDisc[disc]
3    TijăAux ← 1 + 2 + 3 - TijăSursă - TijăDest
4    pentru AltDisc = disc - 1, 1 execută:
5      MutăDisc(AltDisc,TijăAux)
6    sfârșit pentru
7    NumMutări ← rest[(NumMutări + 1)/1000000]
8    TijăDisc[disc] ← TijăDest {mutare directă}
9  sfârșit subalgoritm
10
11 Algoritm Soluție(N, TijăDisc)
12   NumMutări ← 0
13   pentru disc = N - 1, 1, -1 execută:
14     MutăDisc(disc, TijăDisc[N])
15   sfârșit pentru
16 {NumMutări conține acum rezultatul}
17 sfârșit algoritm
```

Pe baza observațiilor de mai sus este evident că acest algoritm generează ordinea corectă a mutărilor. De asemenea, este vizibil că acest lucru încă nu este suficient. Numărul `NumMutări` al mutărilor nu crește niciodată cu mai mult decât 1, deci subalgoritmul `MutăDisc` este apelat o dată pentru fiecare mutare. Deoarece este știut că numărul mutărilor în cazul problemei *turnurilor din Hanoi* crește exponențial cu $O(2^N)$, timpul de execuție va fi cel puțin $O(2^N)$, ceea ce este mult prea mult pentru $N \leq 100.000$.

Așa cum a fost precizat și în enunț, sunt necesare exact 2^i - 1 mutări pentru a muta discurile i, i - 1, ..., 1 de pe o tijă pe alta (aceleași tije pentru toate discurile). Deci, în loc să realizăm aceste 2^i - 1 mutări separat, vom muta direct aceste i discuri pe tija destinație și vom mări numărul mutărilor cu 2^i - 1.

Când mutăm discul i pe tija X, acest lucru îl facem fie deoarece trebuie să eliberăm tija pentru un disc de diame-

tru mai mare, fie deoarece tija X este tija destinație *dest*. În ambele cazuri vrem să mutăm toate discurile mai mici i - 1, i - 2, ..., 1 pe X unul după altul. În subalgoritmul `MutăDisc`, când am ajuns în linia 7, toate aceste discuri mai mici se află pe aceeași tijă. Deoarece acestea vor fi mutate pe `TijăDest` oricum, schimbăm subalgoritmul `MutăDisc` în `MutăDiscuri` cu care le mutăm, și concomitent, folosind ordinea în care se află, calculăm în mod direct numărul mutărilor necesare. Aceasta conduce la următoarea soluție îmbunătățită:

```
1  {mută toate discurile 1, 2, ..., maxDisc pe TijăDest}
2  Subalgoritm MutăDiscuri(maxDisc, TijăDest,
                                    TijăDisc):
3     TijăSursă ← TijăDisc[maxDisc]
4     TijăAux ← 1 + 2 + 3 - TijăSursă - TijăDest
5     dacă maxDisc < 1 atunci
6        *ieșire
7     sfârșit dacă
8     dacă (TijăSursă = TijăDest) atunci
9        MutăDiscuri(maxDisc - 1, TijăDest)
10    altfel
11       MutăDiscuri(maxDisc - 1, TijăAux)
12    sfârșit dacă
13    {acum maxDisc poate fi mutat direct de pe TijăSursă}
14    {pe TijăDest (o mutare) și discurile mai mici pot fi}
15    {mutate de pe TijăAux pe TijăDest folosind 2^maxDisc
16    {- 1 mutări, rezultă un total de 2^maxDisc mutări}
17    pentru i = 1, maxDisc execută:
18       TijăDisc[i] ← TijăDest
19       NumMutări ← rest[(NumMutări +
                  PutereaLuiDoi(disc)) / 1000000]
20    sfârșit pentru
21 sfârșit subalgoritm
22
23 Algoritm Soluție(N, TijăDisc)
24    NumMutări ← 0
25    MutăDiscuri(N - 1, TijăDisc[N])
26 sfârșit algoritm
```

În această versiune îmbunătățită, subalgoritmul `MutăDiscuri` nu se autoapelează decât o singură dată pentru fiecare valoare `maxDisc` egală cu N, N - 1, ..., 1, adică în total de N ori. Pentru fiecare nivel al recursiei timpul de execuție este în medie $O(N)$ datorită ciclului **pentru** din linia 17.

Ideea este de a calcula puterile lui 2 și de a le memora într-un șir, acesta realizându-se în timp constant. Bineînțeles, aceștia se pot calcula modulo 106 pentru a evita întregi mai mari decât cei reprezentabili pe 32 de biți. Astfel, timpul total de execuție al algoritmului este $O(N^2)$, suficient de rapid doar pentru primele 12 teste.

Prin verificarea explicită a faptului că discurile 1, ..., i care trebuie mutate se află sau nu pe aceeași tijă cu un ciclu de tip **for**, se poate realiza un algoritm $O(N^3)$ care rezolvă primele 6-8 teste. (Această soluție nu o vom prezenta).

În continuare vom îmbunătăți algoritmul $O(N^2)$ pentru a realiza unul *liniar*. Pentru a realiza acest lucru vom observa că ciclul din linia 17 este inutil și poate fi omis, deoarece valorile `TijăDisc[]` care se scriu în acest ciclu, *nu se citesc niciodată*. Valorile `TijăDisc[]` sunt citite doar *înainte* de apelurile recursive, deci nu are importanță dacă acestea sunt modificate în timpul sau după aceste apeluri. Deoarece pe noi ne interesează doar numărul mutărilor necesare și știm că în final toate discurile se vor afla pe tija destinație, actualizarea pozițiilor discurilor nu ne interesează și poate fi omisă. Deoarece acest ciclu este singurul care în subalgoritmul `MutăDiscuri` consumă mai mult decât un timp *constant*, și acesta este apelat doar de N ori, avem o soluție $O(N)$. Evident, nici o soluție nu poate fi mai bună (excepție făcând un factor constant), deoarece cu citirea datelor oricum se consumă un timp $O(N)$, iar scrierea rezultatului se face, de asemenea, într-un timp $O(N)$ deoarece numărul care reprezintă soluția este aproximativ egal cu $O(2^N)$, deci are $O(N)$ cifre.

Soluția, care are ordinul de complexitate $O(N)$, este:

```
1  {mută toate discurile 1, 2, ..., maxDisc pe TijăDest}
2  Subalgoritm MutăDiscuri(maxDisc, TijăDest,
                    TijăDisc, PutereaLuiDoi):
3     TijăSursă ← TijăDisc[maxDisc]
4     TijăAux ← 1 + 2 + 3 - TijăSursă - TijăDest
5     dacă maxDisc < 1 atunci
6        *ieșire
7     sfârșit dacă
8     dacă (TijăSursă = TijăDest) atunci
9        MutăDiscuri(maxDisc - 1, TijăDest)
10    altfel
11       MutăDiscuri(maxDisc - 1, TijăAux)
12    sfârșit dacă
13    {acum maxDisc poate fi mutat direct de pe TijăSursă}
14    {pe TijăDest (o mutare) și discurile mai mici pot fi}
15    {mutate de pe TijăAux pe TijăDest folosind 2^maxDisc
16    {- 1 mutări, rezultă un total de 2^maxDisc mutări}
17    pentru i = 1, maxDisc execută:
18       TijăDisc[i] ← TijăDest
19       NumMutări ← rest[(NumMutări +
                  PutereaLuiDoi(disc)) / 1000000]
20    sfârșit pentru
21 sfârșit subalgoritm
22
23 Algoritm Soluție(N, TijăDisc)
24    NumMutări ← 0
25    PutereaLuiDoi[0] ← 1
26    pentru i=1, N execută:
27       PutereaLuiDoi[i] ← rest[(2 *
                  PutereaLuiDoi[i - 1]) / 1000000]
28    sfârșit pentru
25    MutăDiscuri(N - 1, TijăDisc[N],
                                PutereaLuiDoi)
26 sfârșit algoritm
```

P060308: Cursa

Această problemă constă în două subprobleme independente.

Prima subproblemă constă în determinarea numărului total de depășiri. Deoarece toate vitezele navelor sunt constante, după un anumit moment de timp nu vor mai avea loc depășiri, iar navele vor fi sortate în ordinea crescătoare a vitezelor. Două nave având aceeași viteză vor fi ordonate în funcție de pozițiile lor de pornire.

Pentru a efectua sortarea în funcție de viteza navelor, se va folosi algoritmul de sortare *bubblesort*, deoarece este un algoritm stabil și navele având aceeași viteză își vor păstra ordinea după sortare. Datorită restricțiilor impuse asupra fișierului de intrare, adică navele sunt sortate în funcție de poziția de pornire, navele având aceeași viteză, după sortare, vor ocupa același loc.

De fapt, prima subproblemă este echivalentă cu a determina numărul total de interschimbări efectuate de algoritmul de sortare *bublesort*.

Cel mai simplu algoritm constă în a parcurge șirul de nave în ordinea pozițiilor lor în cursă. La fiecare pas se va actualiza numărul de nave care preced poziția curentă și câte dintre acestea vor depăși nava curentă. Această soluție are ordinul de complexitate $O(N \cdot V_{max})$, unde V_{max} reprezintă viteza maximă pentru o navă impusă în enunțul problemei. În continuare prezentăm algoritmul de rezolvare, în pseudocod, pentru această subproblemă:

```
nDepășiri ← 0
pentru i = VMAX, 1, -1 execută:
   nNaveCuV_i ← 0
sfârșit pentru
pentru i = 0, N - 1 execută:
   nNaveCuV_vitezănavai ← nNaveCuV_vitezănavai + 1
sfârșit pentru
pentru i = N - 1, 0, -1 execută:
   nNaveCuV_vitezănavai ← nNaveCuV_vitezănavai - 1
//count passes of ships that are behind position i
   pentru v = viteză_navai + 1, VMAX execută:
      nDepășiri ← nDepășiri + nNaveCuV[v]
   sfârșit pentru
   nDepășiri ← rest[nDepășiri / 1000000];
sfârșit pentru
```

A doua subproblemă constă în a determina primele 10000 de depășiri în ordine cronologică.

În cel mai rău caz au loc $O(N^2)$ depășiri. Un algoritm care compară toate perechile de nave determinând depășirile posibile și sortând aceste depășiri are ordinul de complexitate $O(N^2 \cdot \log N)$ și nu s-ar fi încadrat în timp.

Algoritmul pe care-l vom prezenta în continuare are ordinul de complexitate $O(N + c \cdot \log N)$, unde c reprezintă numărul total de depășiri care trebuie determinate.

Se observă că o navă i poate depăși nava j dacă nava i este prima în spatele navei j și are o viteză mai mare. Ideea de rezolvare constă în a menține o listă de nave și un *heap* care conține, pentru fiecare navă i, nava j care se află imediat în fața ei și momentul de timp când nava i va depăși nava j. Dacă nava i nu poate depăși nava j, atunci acest eveniment nu va fi pus în *heap*. Elementul care se află în vârful *heap*-ului reprezintă următoarea depășire care va avea loc.

La începutul algoritmului, lista de nave se ordonează în funcție de poziția lor la start și se va construi *heap*-ul, elementul din vârful aceastuia reprezentând prima depășire care are loc.

În continuare va trebui să simulăm primele 10000 de depășiri, astfel, la fiecare pas, efectuăm secvențele următoare de instrucțiuni:

- eliminăm din *heap* primul eveniment de forma nava i depășește nava j și îl vom tipări;
- dacă nava i depășește altă navă în viitor, vom insera în *heap* acest eveniment;
- dacă nava care se află imediat în spatele navei j va depăși nava i (care va depăși nava j înainte de a depăși nava i) va trebui să actualizăm acest eveniment, lucru care se realizează prin urcarea acestuia în *heap*.

Algoritmul de rezolvare a acestei probleme în pseudocod este următorul:

```
sortează(nave)
pentru i = 1, N - 1 execută:
   dacă nave_i depășește nave_i+1 atunci
      pune eveniment în heap
   sfârșit dacă
sfârșit pentru
pentru i = 1, 10000 execută:
   dacă heap-ul este gol atunci
      *întrerupe ciclu
   sfârșit dacă
   scoate primul eveniment din heap
   tipărește eveniment
   interschimbă navele eveniment.i și eveniment.j
   dacă nave_eveniment.i depășește nave_eveminent.i+1 atunci
      pune acest eveniment în heap;
   sfârșit dacă
   dacă nave_eveniment.j-1 depășește nave_eveminent.j atunci
      actualizează evenimentul navei j
   sfârșit dacă
sfârșit pentru
```

P060309: Pătrat

Algoritmul de rezolvare pentru această problemă, pe care-l vom prezenta în continuare, nu efectuează mai mult de $3 \cdot N$ interogări, și astfel se încadrează între limite.

Mai întâi, trebuie să observăm că fiecare drum dintre nodul (1, 1) și un nod v dat are aceeași lungime.

Notăm cu $D(v)$ distanța dintre (1, 1) și v. Dacă $D(v) > L$, știm că nu pot fi soluții nici nodurile din dreapta lui v, nici cele situate sub v, deoarece distanțele până la nodul (1, 1) sunt mai mari.

Vom începe explorarea grafului pornind de la nodul (1, 1) și ne vom muta în jos până la atingerea unui nod v pentru care $D(v) \geq L$. Dacă $D(v) = L$, am găsit o soluție. Dacă $D(v) > L$, știm că nici nodul curent și nici unul dintre nodurile aflate în dreapta nodului curent sau sub el, nu pot fi soluții. Mai mult, știm că distanța până la nodul aflat deasupra este prea mică. Astfel, vom trece la nodul din dreapta-sus.

Acum vom prelucra iterativ vecinul din dreapta, de fiecare dată când distanța dintre nodul curent și acesta este prea mică sau prelucrăm vecinul de deasupra dacă distanța este prea mare. De fiecare dată se poate extrage un grup de noduri din mulțimea soluțiilor posibile, așa cum se poate vedea în figura 1.

Dacă trebuie să avansăm într-o direcție în care nu mai există noduri, putem trage concluzia că pentru acest test nu există soluție.

Am uitat un caz: Ce se întâmplă atunci când nu putem ajunge, mutându-ne în jos, la un nod față de care distanța este mai mare sau egală cu L? Răspunsul este simplu: vom continua mutările în dreapta și vom proceda ca mai sus până când găsim un nod v pentru care avem $D(v) > L$.

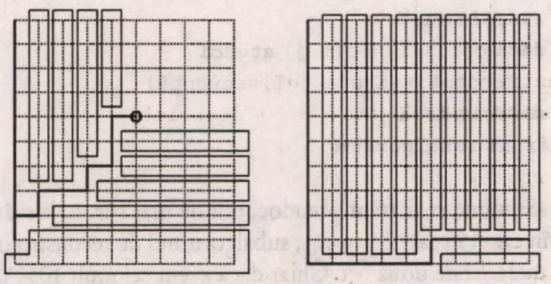


Figura 1

Afirmația de la începutul analizei este adevărată, adică nu vom executa decât cel mult $3 \cdot N$ prelucrări, deoarece ne vom deplasa în jos de cel mult N - 1 ori, ne vom deplasa în dreapta de cel mult N - 1 ori și în sus de cel mult N - 1 ori.

P060310: Colierul de perle

Problema constă în găsirea mutărilor care conduc la câștig. O mutare conduce la câștig, dacă jucătorul mută optim în continuare.

Dacă presupunem că ambii jucători joacă optim, mutările fie conduc la câștig, fie la pierderea jocului din moment ce în acest joc nu există remiză.

În plus, o mutare conduce la câștig, dacă un pitic poate da șiragul de mărgele unui alt pitic din tribul lui care are la dispoziție o mutare de câștig, sau dacă el poate da șiragul de mărgele unui alt pitic din celălalt trib care nu are la dispoziție o mutare de câștig.

O stare a jocului este determinată de starea șiragului de mărgele și a piticului curent:

```
Subalgoritm AreMutaredeCâștig(c1c2...ck,p)
  returnează (∃p' ∈ listp,c1 astfel încât (culoarep =
  culoarep' și AreMutaredeCâștig(c2...ck,p'))
  sau (culoarep ≠ culoarep' și nu
  AreMutaredeCâștig(c2...ck,p')))
sfârșit subalgoritm.
```

Datorită restricțiilor impuse în enunțul problemei, `AreMutaredeCâștig(ck,p')` returnează întotdeauna valoarea **adevărat** deoarece, la sfârșitul recursiilor, în listă rămâne numai diamantul, iar `p'` este sigur un pitic de culoare verde.

În subalgoritmul anterior, $list_{p,c}$ reprezintă lista albă pentru piticul `p`, dacă `c` are valoarea `1` și lista neagră pentru piticul `p`, dacă `c` are valoarea `0`, iar $culoare_p$ reprezintă culoarea piticului `p`.

Pe baza acestor observații se pot determina ușor mutările care conduc la câștig folosind metoda programării dinamice într-o abordare *bottom-up*.

Pentru a determina mutările care duc la câștig vom folosi un tablou bidimensional auxiliar de dimensiune $L \times N$ iar fiecare element $c_{i,p}$ al acestuia va avea valoarea 0 dacă, pornind din starea curentă (s-au efectuat i-1 mutări), piticul p nu are mutare de câștig și valoarea 1 dacă, pornind din starea curentă, piticul p are mutare de câștig.

Acest tablou se construiește traversând indicii din șirag în ordine inversă, adică de la L la 1.

În continuare, având construit tabloul c, la fiecare mutare a unui pitic verde se va alege un pitic de pe lista acestuia, indicată de mărgeaua la care s-a ajuns, astfel încât să fie verificată condiția impusă în cadrul subalgoritmului prezentat anterior, adică fie un pitic verde care să aibă mutare de câștig, fie un pitic roșu care să nu aibă mutare de câștig.

P060311: Registru de deplasare

Ideea de rezolvare a acestei probleme constă în a reprezenta poarta *XOR* ca o funcție liniară cu valori în mulțimea $Z_2 = \{0, 1\}$. Plecând de la această idee vom avea de rezolvat un sistem liniar cu N ecuații și N necunoscute folosind metoda lui *Gauss*.

Observăm că, în mulțimea Z_2, înmulțirea a două numere este echivalentă cu aplicarea operatorului binar **ȘI** între numere, iar adunarea este echivalentă cu aplicarea operatorului binar **SAU-Exclusiv**.

Această observație ne permite să descriem efectul combinat al comutatoarelor s_i și al porții *XOR* folosind următoarea expresie liniară:

$$s_1 \cdot a_1 + s_2 \cdot a_2 + \ldots + s_N \cdot a_N.$$

Fie o_i ($i \geq 1$) cel de-al i-lea bit transmis de poarta *XOR* (care reprezintă de fapt cea de-a i-a valoare de pe cea de-a doua linie a fișierului de ieșire). Primii N biți transimiși de poarta *XOR*, o_1, o_2, ..., o_N, reprezintă valorile inițiale ale biților registrului de deplasare a_1, a_2, ..., a_N. Primul bit transmis care necesită calcule este o_{N+1}. Când o_{N+1} este calculat, biții a_i ai registrului de deplasare sunt ocupați (în

ordine inversă) de valorile o_1, o_2, ..., o_N. Pe baza expresiei liniare construite anterior, obținem prima ecuație a sistemului:

$$o_{N+1} = s_N \cdot o_1 + s_{N-1} \cdot o_2 + ... + s_1 \cdot o_N.$$

Pentru a calcula valorile o_{N+2}, o_{N+3}, ..., $o_{2\cdot N}$ se vor construi ecuații similare. Vom inversa ordinea comutatoarelor definind $s'_i = s_{n-i+1}$ pentru a putea manipula mai ușor aceste ecuații. Astfel, avem de rezolvat sistemul de ecuații:

$$s'_1 \cdot o_1 + s'_2 \cdot o_2 + ... + s'_N \cdot o_N = o_{N+1}$$
$$s'_1 \cdot o_2 + s'_2 \cdot o_3 + ... + s'_N \cdot o_{N+1} = o_{N+2}$$
$$...$$
$$s'_1 \cdot o_N + s'_2 \cdot o_{N+1} + ... + s'_N \cdot o_{2\cdot N-1} = o_{2\cdot N}$$

În continuare vom construi matricea M de dimensiune $N \times N + 1$ care va conține coeficienții ecuațiilor și termenii liberi.

Rezolvarea sistemului de ecuații constă în a obține valoarea 0 sub diagonala principală prin transformări liniare ale liniilor și interschimbări ale acestora și reconstituirea valorilor s'_i în ordine inversă conform metodei lui *Gauss* al cărei ordin de complexitate este $O(N^3)$.

Dacă sistemul de ecuații nu are soluție, atunci se va scrie în fișierul de ieșire valoarea -1 și în caz contrar se vor scrie valorile s'_i, $1 \le i \le N$, în ordine inversă.

P060312: Călătoria

Problema cere să se afișeze toate cele mai lungi subsecvențe comune ale două șiruri de caractere. Prima idee care ar putea fi fructificată constă în utilizarea *metodei programării dinamice*, în forma ei standard, pentru a determina lungimea maximă a subsecvențelor comune, urmând ca apoi să se construiască, pe rând, fiecare astfel de subsecvență.

Fie $A = (a_1, a_2, ..., a_n)$ primul șir și $D = (b_1, b_2, ..., b_m)$ cel de-al doilea șir. Fie $c_{i,j}$ lungimea celei mai lungi subsecvențe comune ale șirurilor $(a_1, a_2, ..., a_i)$ și $(b_1, b_2, ..., b_j)$. Atunci:

$$c_{i,j} = \begin{cases} 0, & \text{dacă } i = 0 \text{ sau } j = 0 \\ c_{i-1,j-1} + 1, & \text{dacă } i,j > 0 \text{ și } a[i] = b[j] \\ \max(c_{i-1,j}, c_{i,j-1}), & \text{dacă } i,j > 0 \text{ și } a[i] \neq b[j] \end{cases}.$$

Lungimea celei mai lungi subsecvențe comune a șirurilor A și B este dată de $c_{n,m}$.

Pe baza acestor recurențe algoritmul de programare dinamică este următorul:

```
...
pentru i = 1, n execută:
  c[i,0] ← 0
sfârșit pentru
pentru j=0, m execută:
  c[0,j] ← 0
sfârșit pentru
pentru i=1, n execută:
  pentru j=1, m execută:
    dacă a[i] = b[j] atunci
      c[i,j] ← c[i-1,j-1]+1
    altfel
      c[i,j] ← Max(c[i-1,j],c[i,j-1])
    sfârșit dacă
  sfârșit pentru
sfârșit pentru
...
```

În scopul construirii celei mai lungi subsecvențe, trebuie să reconstruim rezultatul pe baza tabloului *c*. Această reconstituire se realizează cu algoritmul:

```
Subalgoritm Reconstituire(i,j,secvență):
  dacă i = 0 sau j = 0 atunci
    adaugă secvență mulțimii celor mai lungi
                                    subsecvențe
    *ieșire
  sfârșit dacă
  dacă a[i] = b[j] atunci
    alipește a[i] secvenței
    Reconstituire(i-1,j-1,secvență)
    *ieșire
  sfârșit dacă
  dacă c[i-1,j] = c[i,j] atunci
    Reconstituire(i-1,j,secvență)
  sfârșit dacă
  dacă c[i,j-1] = c[i,j] atunci
    Reconstituire(i,j-1,secvență)
  sfârșit dacă
sfârșit subalgoritm
```

Studiind mai atent pseudocodul de mai sus, observăm că în cazul în care $c_{i-1,j} = c_{i,j-1}$, subalgoritmul de reconstituire se apelează de două ori. Chiar dacă avem cel mult 1000 de cele mai lungi subsecvențe comune, acest lucru nu spune nimic despre numărul de posibilități în care acestea se pot construi. Modalitatea de reconstituire de mai sus va încerca toate posibilitățile de a construi toate cele mai lungi subsecvențe comune. De exemplu, dacă avem două șiruri de caractere "aaaaaaabccccccd" și "abbbbbbbcddddddd" există 1778966 posibilități de a construi singura cea mai lungă subsecvență comună "abcd".

Cum am putea evita construirea aceleiași secvențe de mai multe ori? Am putea să le construim în paralel cu determinarea lungimilor. Avem nevoie de un tablou de mulțimi în care să păstrăm toate cele mai lungi subsecvențe comune ale șirurilor $(a_1, a_2, ..., a_i)$ și $(b_1, b_2, ..., b_j)$.

Algoritmul bazat pe programare dinamică acum arată în felul următor:

```
...
pentru i = 1, n execută:
  c_{i,0} ← 0
  mulțime_{i,0} ← ∅
sfârșit pentru
pentru j = 0, m execută:
  c[0,j] ← 0
```

```
    mulțime[0,j] ← ∅
  sfârșit pentru
  pentru i = 1, n execută:
    pentru j=1, m execută:
      dacă a_i = b_j atunci
        c_{i,j} ← c_{i-1,j-1} + 1
        mulțime[i,j] ← mulțime[i-1,j-1]
        alipește a_i la sfârșitul tuturor secvențelor din
                                          mulțime_{i,j}
      altfel
        c_{i,j} ← Max(c_{i-1,j}, c_{i,j-1})
        dacă c_{i-1,j} > c_{i,j-1} atunci
          mulțime_{i,j} ← mulțime_{i-1,j}
        altfel
          dacă c_{i,j-1} > c_{i-1,j} atunci
            mulțime_{i,j} ← mulțime_{i,j-1}
          altfel
            mulțime_{i,j} ← mulțime_{i-1,j} ∪ mulțime_{i,j-1}
      sfârșit dacă
    sfârșit dacă
  sfârșit dacă
 sfârșit pentru
sfârșit pentru
...
```

A mai rămas să analizăm dimensiunea spațiului de memorare, necesară tabloului de mulțimi. Există cel mult 1000 de cele mai lungi subsecvențe comune având lungimea maximă egală cu 80, și astfel dimensiunea tabloului este 80 × 80. Acesta ar necesita 512 *MB* dacă s-ar folosi memoria statică, sau un pic mai puțină dacă în loc de un tablou alocat static s-ar folosi o listă înlănțuită, alocată dinamic. De altfel, cu tabloul plin s-ar fi rezolvat nouă din cele 10 teste, folosind memoria pusă la dispoziție, având dimensiunea 16 *MB*.

În continuare observăm că în ciclul "**pentru** i = 1, n", la un moment dat, numai linia curentă și cea precedentă sunt necesare în timpul prelucrării. Astfel este posibil să se folosească un tablou de dimensiune 2 × 80 și mulțimile vor fi indexate în tablou cu i *modulo* 2. Astfel forma finală a algoritmului este:

```
...
pentru i = 1, n execută:
  c[i,0] ← 0
sfârșit pentru
pentru j =0, m execută:
  c_{0,j} ← 0
  mulțime_{0,j} ← ∅
  mulțime_{1,j} ← ∅
sfârșit pentru
pentru i = 1, n execută:
  pentru j=1, m execută:
    dacă a_i = b_j atunci
      c_{i,j} ← c_{i-1,j-1} + 1
      mulțime_{i mod 2,j} ← mulțime_{1-(i mod 2),j-1}
      alipește a_i la sfârșitul tuturor secvențelor din
                                        mulțime_{i mod 2,j}
    altfel
      c_{i,j} ← Max(c_{i-1,j}, c_{i,j-1})
      dacă c_{i-1,j} > c_{i,j-1} atunci
        mulțime_{i mod 2,j} ← mulțime_{1-(i mod 2),j}
      altfel
        dacă c_{i,j-1} > c_{i-1,j} atunci
          mulțime_{i mod 2,j} ← mulțime_{i mod 2,j-1}
        altfel
          mulțime_{i mod 2,j} ← mulțime_{1-(i mod 2),j} ∪
                                  mulțime_{i mod 2,j-1}
      sfârșit dacă
    sfârșit dacă
  sfârșit dacă
 sfârșit pentru
sfârșit pentru
Afișează toate secvențele din mulțime_{n mod 2,m}
...
```

Astfel am realizat o rezolvare și mai rapidă. Ideea este de a folosi o versiune modificată a algoritmului de reconstituire prezentat.

Vom încerca să construim doar acele cele mai lungi subsecvențe comune x în care x_i reprezintă prima apariție după litera x_{i-1} în ambele șiruri de caractere. Celelalte construcții nu ar conduce la altă cea mai lungă subsecvență comună. Pentru a obține rezultatul cerut, aplicăm căutare pe baza traversării în lățime.

Subalgoritmul de reconstituire este următorul:

```
Subalgoritm Reconstituire2(i,j,secvență):
  dacă i = 0 sau j = 0 atunci
    adaugă secvență mulțimii celor mai lungi
                                    subsecvențe
    *ieșire
  sfârșit dacă
  inserează (i,j) în coadă
  cât timp coada nu este vidă execută:
    (k,l) ← primul element din coadă
    dacă a_k = b_l atunci
      dacă Reconstituire2 nu a fost apelat pentru
      litera a_k alipită în fața lui secvență atunci
        alipește a_k la secvență
        Reconstituire2(i-1,j-1,secvență)
      altfel
        dacă c_{k-1,l} = c_{k,l} și (k-1,l) nu este deja în
        coadă atunci
          adaugă (k-1,l) în coadă
        sfârșit dacă
        dacă c_{k,l-1} = c_{k,l} și (k,l-1) nu este deja în
        coadă atunci
          adaugă (k,l-1) în coadă
        sfârșit dacă
      sfârșit dacă
    sfârșit dacă
  sfârșit cât timp
sfârșit subalgoritm
```

Rundele #06 - #15

În cadrul acestui articol vom publica detalii despre desfășurarea rundelor 06-15 ale ediției 2003/2004 a concursului de programare Bursele Agora organizat de revista noastră.

P010416: Arena din Erathia

Vom împărți coordonatele creaturilor în mai multe mulțimi, fiecare dintre aceste mulțimi conținând coordonate ale unor creaturi care fac parte din aceeași rasă.

Pentru fiecare dintre aceste mulțimi vom determina cea mai îndepărtată pereche de puncte. Cea mai defavorizată rasă va fi cea pentru care distanța dintre cele mai îndepărtate puncte va fi maximă.

Cea mai simplă metodă de rezolvare a acestei probleme constă în determinarea distanțelor dintre toate perechile de puncte (vor fi $N \cdot (N - 1) / 2$ astfel de perechi) și alegerea celei mai mari dintre ele. Am notat cu N numărul punctelor dintr-o mulțime.

Așadar, problema se reduce la o simplă determinare a unei valori maxime dintre $N \cdot (N - 1) / 2$, valori care sunt calculate. Evident, ordinul de complexitate al acestui algoritm este $O(N^2)$.

Din nefericire, acest ordin de complexitate nu este acceptabil pentru această problemă, deoarece putem avea până la 5000 de puncte.

Pentru a obține un algoritm eficient, vom observa mai întâi faptul că puncte le care formează cea mai îndepărtată pereche se află, obligatoriu, pe înfășurătoarea convexă a mulțimii de puncte.

Vom demonstra această afirmație prin *metoda reducerii la absurd*. Să presupunem că unul dintre cele două puncte nu se află pe înfășurătoarea convexă, așadar se află în interiorul acesteia. Vom nota acest punct prin Q, iar celălalt punct prin Q. Dacă vom prelungi segmentul care unește punctele P și Q, atunci acesta va intersecta o latură a înfășurătorii convexe sau va trece printr-un alt punct al acesteia.

În acest din urmă caz se observă imediat că distanța de la P la Q este mai mică decât distanța de la P la punctul respectiv, deci punctele P și Q nu se pot afla la distanța maximă.

Pentru primul caz vom considera că latura intersectată are ca extremități punctele R și S și prelungirea segmentului care unește punctele P și Q intersectează latura respectivă în punctul T.

Pentru început se observă faptul că lungimea segmentului care unește punctele P și T este mai mare decât cea care unește punctele P și Q.

Vom considera acum triunghiul PRS (determinat de vârfurile P, R și S). Punctul T se va afla pe latura RS a acestui triunghi. Există o teoremă care afirmă că cel puțin una dintre laturile PR și PS are lungimea mai mare decât segmentul PT. Ca urmare una dintre laturile triunghiului va fi mai lungă decât segmentul PT care, la rândul său, este mai lung decât segmentul PQ.

Datorită faptului că această latură are ca extremități două vârfuri ale înfășurătorii convexe, deducem imediat că punctele P și Q nu se pot afla la distanța maximă.

În concluzie, primul pas necesar pentru a determina punctele aflate la distanța maximă este determinarea unei înfășurători convexe.

O variantă care poate fi utilizată în continuare este considerarea succesivă a vârfurilor înfășurătorii și determinarea, printr-o metodă similară cu căutarea binară, a celui mai îndepărtat vârf față de vârful considerat.

Se va considera nodul din stânga (sau dreapta) nodului ales. Atâta timp cât distanța crește, la fiecare pas k, vom "sări" la al 2^k-lea nod. În momentul în care distanța începe să scadă, vom reveni la nodul anterior și vom începe să înjumătățim "distanța" saltului.

Dacă am ajuns la pasul l, acum, la fiecare, pas vom descreste cu 1 puterea. Dacă la un anumit pas, puterea este p, vom efectua un salt la al 2^p-lea nod. Dacă distanța descrește vom reveni, iar dacă nu, vom continua de la vârful respectiv.

În final, vom obține vârful aflat la distanța maximă față de nodul considerat. După considerarea tuturor vârfurilor, vom obține distanța maximă între două puncte din mulțimea dată.

Algoritmul descris funcționează corect deoarece, parcurgând nodurile într-un anumit sens distanța începe să crească până la un moment dat, după care ea începe să scadă.

O a doua variantă constă în alegerea unui vârf P_1 al înfășurătorii și parcurgerea acesteia (începând cu punctul din

stânga sau din dreapta sa) atâta timp cât distanța crește. Vom nota punctul respectiv prin P. În continuare vom trece la punctul P_2 (cel aflat pe înfășurătoare imediat lângă punctul P_1, în funcție de sensul ales) și vom continua parcurgerea.

Poate fi ușor observat faptul că nu are sens să mai verificăm toate punctele parcurse la pasul anterior (cele până la P) deoarece distanțele obținute vor fi mai mici. Așadar, vom continua de la punctul P atâta timp cât distanța va crește.

Vom trece apoi la punctul P_3 și vom repeta în continuare procedeul până în momentul în care am luat în considerare toate punctele de pe înfășurătoare.

Evident, în final vom cunoaște distanța maximă dintre două puncte de pe înfășurătoare și, implicit, distanța maximă dintre două puncte ale mulțimii date.

Determinarea înfășurătorii convexe are ordinul de complexitate $O(N \cdot \log N)$, dacă utilizăm scanarea *Graham* sau $O(N \cdot h)$, dacă utilizăm potrivirea *Jarvis*. (Am notat prin h numărul punctelor aflate pe înfășurătoarea convexă a mulțimii de puncte.)

În situația în care, după determinarea înfășurătorii, vom folosi metoda de căutare binară, distanța maximă va fi determinată într-un timp de ordinul $O(h \cdot \log h)$, datorită faptului că vom efectua h căutări binare, fiecare având ordinul de complexitate $O(\log h)$.

În cazul în care vom utiliza cea de-a doua metodă, aceasta va avea ordinul de complexitate $O(h)$, deoarece se poate observa faptul că prin avansări se va parcurge de cel mult două ori înfășurătoarea.

Ca urmare, ordinul de complexitate al algoritmului de determinare a unei perechi de puncte aflate la distanță maximă este același cu cel al algoritmului folosit pentru determinarea înfășurătorii convexe.

Au rezolvat corect...

Dintre cei 81 de participanți la etapa a șasea a concursului, 17 au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Adrian-Romulus Ciripan, România
Adrian Diaconu, România
Silviu-Ionuț Gănceanu, România
Ivan Kazmenko, Federația Rusă
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Mihai Moalfă, România
Cosmin Molea, România
Andrei Păduraru, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Claudiu Rad, România
Andrei-Ionuț Tudor, România
Sorin Vultureanu, România

P010417: Merlin

Problema s-a dovedit destul de dificilă, dovadă fiind numărul mic al concurenților care au reușit să obțină punctajul maxim.

O posibilă abordare, care ar fi dus la obținerea punctajului maxim, constă în folosirea *metodei greedy* pentru a obține o configurație care să conțină un număr cât mai mare de pietricele.

După obținerea acestei configurații, ea putea fi îmbunătățită cu ajutorul metodei *backtracking*.

Au rezolvat corect...

Dintre cei 58 de concurenți care au participat la această etapă a concursului, doar patru au reușit să obțină punctajul maxim:

Mugurel-Ionuț Andreica, România
Sergiu Hlihor, România
Daniel Păsăilă, România
Csaba Pătcaș, România

P010418: Distribuție

Pentru a identifica orașele deservite de prima tipografie (respectiv de a doua) vom determina, mai întâi, distanțele minime de la prima tipografie (orașul identificat prin 1) la toate orașele, respectiv de la a doua tipografie (orașul identificat prin 2) la toate orașele.

În acest scop putem utiliza un algoritm pentru determinarea drumurilor minime de la un nod al unui graf la toate celelalte (*Dijkstra*). Am considerat orașele ca fiind nodurile unui graf neorientat ale cărui muchii reprezintă străzile dintre orașe, iar costurile muchiilor sunt date de lungimile străzilor.

Pentru fiecare oraș i vom cunoaște distanțele d_{1i} și d_{2i} față de cele două tipografii.

Vom calcula valorile $d_i = d_{1i} - d_{2i}$ pentru toate orașele și vom sorta orașele în ordine crescătoare în funcție de aceste valori.

Primele $N / 2$ orașe (în ordinea determinată) vor fi deservite de prima tipografie, iar celelalte orașe de către cea de-a doua.

Au rezolvat corect...

Dintre cei 43 de participanți la cea de-a opta etapă, aproape jumătate (18) au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Andrei Blaj, România
Iulian Blajuț, România
Adrian Diaconu, România
Ioan-Dan Docleanu, România
Silviu-Ionuț Gănceanu, România
Andrei-Gabriel Ion, România
Ivan Kazmenko, Federația Rusă
Vasanth Jai, India
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Mihai Moalfă, România
Dalian Valeriu Moraru, România
Mircea Pașoi, România
Daniel Păsăilă, România
Alexandru Peptan, România
Victor Podeanu, România
Tudor-Alexandru Rabaea, România

P010419: Cerc

Această problemă este foarte asemănătoare cu problema *Petrecere nebunatică* propusă spre rezolvare la etapa regională a concursului *ACM* din acest an. Rezolvarea acestei probleme a fost prezentată în *GInfo* 13/8 (decembrie 2003).

Au rezolvat corect...

Dintre cei 57 de participanți la cea de-a noua etapă a concursului, doar cinci au obținut punctajul maxim:

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Bogdan Nicolae, România
Csaba Pătcaș, România
Valentin Stanciu, România

P010420: Explozie

Problema se reduce la a determina numărul punctelor de coordonate întregi aflate în interiorul unei sfere de rază dată.

Pentru fiecare punct de coordonate întregi pozitive (x, y, 0) aflate în interiorul sferei vom determina cel mai mare număr întreg z astfel încât punctul de coordonate (x, y, z) se află în interiorul sferei.

Valoarea z se obține rezolvând ecuația $x^2 + y^2 + z^2 = r^2$, unde r este raza sferei. Vom considera partea întreagă $[z]$ a soluției pozitive a acestei ecuații.

În cazul în care valorile x și y sunt nenule, vom adăuga valoarea $4 \cdot [z] + 2$ la numărul de puncte din interiorul sferei. Dacă doar una dintre valorile x și y este nulă, numărul punctelor adăugate va fi $2 \cdot [z] + 1$. În situația în care ambele valori sunt nule, valoarea adăugată este $[z] + 1$.

Au rezolvat corect...

La cea de-a zecea etapă a concursului nostru au participat 43 de concurenți. Dintre aceștia doar trei au obținut numărul maxim de puncte:

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Ivan Kazmenko, Federația Rusă

P010421: Recunoașterea caracterelor

Pentru a compara litera care trebuie recunoscută cu o literă dată vom construi un graf bipartit. Nodurile din prima mulțime vor fi date de elementele cu valoarea 1 din matricea corespunzătoare primei litere, iar nodurile din a doua mulțime vor fi date de elementele cu valoarea 1 din matricea corespunzătoare celei de-a doua litere.

Fiecare nod din prima mulțime va fi unit printr-o muchie de toate nodurile din cea de-a doua. Costurile muchiei corespunzătoare unei perechi de noduri este dată de distanța *Manhattan* dintre punctele corespunzătoare, iar capacitatea sa este 1.

În acest graf bipartit vom determina un cuplaj maxim de cost minim. Costul acestui cuplaj reprezintă distanța dintre matrice.

Pentru a ne asigura că este respectată condiția care impune faptul ca unui element dintr-o matrice să îi corespundă $[p / q]$ sau, eventual, $[p / q] + 1$ puncte din cealaltă, capacitățile muchiilor care vor uni nodul sursă (cel introdus pentru determinarea cuplajului cu ajutorul fluxului maxim) de nodurile din prima mulțime, vor fi date de partea întreagă superioară a raportului p / q. Valorile p și q au semnificația din enunț. După determinarea tuturor cuplajelor maxime de cost minim, rezultatul va fi litera corespunzătoare cuplajului cu cel mai mic cost.

Au rezolvat corect...

Doar 36 de participanți au trimis soluții la această rundă. Dintre aceștia, cinci au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Ivan Kazmenko, Federația Rusă
Kiril Minkov, Bulgaria
Csaba Pătcaș, România
Alexandru Peptan, România

P010422: Ciobănașul

Pentru a determina numărul pătratelor care au în vârfuri parcele nivelate vom considera toate perechile de linii.

Pentru fiecare pereche vom determina numărul k al coloanelor care conțin valoarea 1 în ambele linii. În acest scop putem utiliza conjuncția logică pentru a mări viteza de execuție. După determinarea valorii k, vom ști că pe cele două linii vom avea $k \cdot (k - 1) / 2$ pătrate.

Vom însuma valorile corespunzătoare fiecărei perechi de linii și vom afișa rezultatul.

Au rezolvat corect...

La această rundă au participat 48 de concurenți dintre care cinci au obținut punctajul maxim:

Chattopadhyay Amit, India
Adrian Cârcu, România
Silviu-Ionuț Gănceanu, România
Vasanth Jai, India
Mihail-Cosmin Pițu-Rada, România

P010423: Joc

Pentru început vom observa că o grămadă care conține un număr de elemente care este multiplu de 4 poate fi "câștigată" de al doilea jucător.

Indiferent care este numărul de pietre pe care îl alege primul jucător (acesta nu va putea lua toate pietrele deoarece numărul total nu este prim, fiind un multiplu de 4), cel de-al doilea jucător va putea lua 1, 2 sau 3 pietre pentru ca numărul pietrelor rămase să fie din nou multiplu de patru. Evident, după mutarea primului jucător numărul rămas nu va fi multiplu de 4, deoarece numărul pietrelor luate trebuie să fie 1 sau un număr prim (deci nu poate fi multiplu de 4).

Așadar, strategia primului jucător este ca să obțină o configurație în care toate grămezile să conțină un număr de pietre care este multiplu de 4.

În acest scop se va utiliza strategia pentru jocul *NIM*, dar considerând doar resturile împărțirii la 4 ale numerelor de pietre din grămezi.

Primul jucător va avea o strategie sigură de câștig doar dacă are o strategie pentru această variantă "redusă" a jocului *NIM*.

Au rezolvat corect...

La această etapă au participat 31 de concurenți, dintre care 13 au obținut punctajul maxim.

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Haryanto Lego, Statele Unite ale Americii
Mihai-Vlad Pantiș, România
Mircea Pașoi, România
Csaba Pătcaș, România
Alexandru Peptan, România
Mihail-Cosmin Pițt-Rada, România
Adrian Vladu, România

P010424: Cifre

Această problemă poate fi foarte ușor rezolvată folosind *principiul cutiei lui Dirichlet*.

Vom concatena numărul format din cifrele date cu el însuși până în momentul în care restul împărțirii la n va fi 0 sau obținem un rest care a mai fost obținut la un pas anterior. Pe baza principiului, suntem siguri că vom ajunge în una dintre cele două situații după cel mult n pași.

Dacă am obținut restul 0, avem un rezultat. Dacă nu, rezultatul va fi dat prin simpla scădere a celor două numere care duc la obținerea aceluiași rest prin împărțirea la n. Dacă cele două resturi se obțin la iterațiile p și q, atunci diferența constă în simpla înlocuire a ultimelor $k \cdot (q - p)$ cifre ale rezultatului cu cifra 0 (care face parte cu siguranță dintre cifrele date).

Există un mic amănunt care poate duce la o mică greșeală. Rezultatul nu poate să înceapă cu cifra 0. Pentru a evita această situație este suficient ca, în șirul cifrelor date, să nu punem cifra 0 pe prima poziție.

Au rezolvat corect...

Douăzeci dintre cei 54 de participanți la această etapă au reușit să obțină numărul maxim de puncte.

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Eduard-Gabriel Băzăvan, România
Liviu Ciortea, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Ivan Kazmenko, Federația Rusă
Vasanth Jai, India
Haryanto Lego, Statele Unite ale Americii
Mihai Moalfă, România
Marius Nicolae, România
Mihai-Vlad Pantiș, România
Mircea Pașoi, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Mihail-Cosmin Pițt-Rada, România
Victor Podeanu, România
Sorin Stancu-Mara, România
Adrian Vladu, România

P010425: Stația spațială

Aceasta a fost cea mai simplă problemă dintre cele propuse spre rezolvare la primele 15 runde. Mai mult de trei sferturi dintre cei care au trimis soluții au reușit să o rezolve corect. Soluția acestei probleme este foarte simplă; se utilizează o strategie *greedy* care constă în considerarea primei nave (*USS Enterprise*) și eliminarea tuturor celor care urmează și se află la o distanță prea mică. Se trece la următoarea navă și se elimină din nou navele aflate la distanță prea mică. Procedeul continuă până în momentul în care se ajunge la ultima navă.

Au rezolvat corect...

Dintre cei 52 de concurenți care au participat la această rundă, 41 au obținut punctajul maxim:

Renald, Indonezia
Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Eduard-Gabriel Băzăvan, România
Iulian Blajuț, România
Adrian Cârcu, România
Dragoș Cavulea, România
Cristian Cernat, România
Liviu Ciortea, România
Adrian-Romulus Ciripan, România
Arthur Costea, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Adrian Ghiuță, România
Andrei Gönczi, România
Sergiu Hlihor, România
Ing Ing, Indonezia
Kazmenko Ivan, Federația Rusă
Vasanth Jai, India
Andrei Janca, România
Johannes Slotta, Germania
Stelian Kelemen, România
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Marius Nicolae, România
Mircea Pașoi, România
Ciprian Păduraru, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Alexandru Peptan, România
Vlad Petcu, România
Mihail-Cosmin Pițt-Rada, România
Alexandru Popa, România
Claudiu Rad, România
Alexandru Stan, România
Valentin Stanciu, România
Sorin Stancu-Mara, România
Andrei-Ionuț Tudor, România
Adrian Vladu, România
Sorin Vultureanu, România

Bursele Agora 2003/2004 Top 50

1.	Mugurel-Ionuț Andreica, București	1420
2.	Ivan Kazmenko, Federația Rusă	1300
3.	Adrian Cârcu, Bistrița	1167
4.	Chattopadhyay Amit, India	1147
5.	Kiril Minkov, Bulgaria	1137
6.	Daniel Păsăilă, Suceava	1045
7.	Silviu-Ionuț Gănceanu, Bârlad	1038
8.	Mircea Pașoi, Ploiești	1036
9.	Lego Haryanto, Statele Unite ale Americii	1030
10.	Liviu Ciortea, București	969
11.	Jai Vasanth, India	932
12.	Csaba Pătcaș, Oradea	845
13.	Peptan Alexandru, Dr. Tr. Severin	805
14.	Mihai-Vlad Pantiș, Cluj-Napoca	804
15.	Adrian Vladu, București	790
16.	Mihail-C. Pițt-Rada, Dr. Tr. Severin	763
17.	Sergiu Hlihor, Lugoj	749
18.	Adrian Diaconu, Bucuresti	704
19.	Marius Dumitran, București	692
19.	Claudiu Rad, Deva	692
21.	Valentin Stanciu, București	622
22.	Alexandru Popa, București	600
23.	Ioan-Dan Docleanu, Nădlac	580
24.	Victor Podeanu, București	550
25.	Sorin Stancu-Mara, București	532
26.	Tudor-Alexandru Rabaea, Constanța	497
27.	Mihai Moalfă, Sîngeorz-Băi	492
28.	Andrei-Ionuț Tudor, Slobozia	487
29.	Andrei Gönczi, Arad	477
30.	Dalian-Valeriu Moraru, București	469
31.	Adrian-Romulus Ciripan, Alba-Iulia	430
32.	Irinel Besliu, Vaslui	413
33.	Iulian Blajuț, Luncani	402
34.	Cosmin Matei, Ploiești	380
35.	Andrei Janca, Dr. Tr. Severin	364
36.	Călin Bindea, Bistrița	343
37.	Bogdan Nicolae, Sibiu	319
38.	Ciprian Păduraru, Bârlad	318
39.	Dragoș Căvulea, Botoșani	303
40.	Silvestru-Cosmin Negrușeri, Bistrița	300
41.	Andrei Blaj, Suceava	275
41.	Liviu-Laurențiu Iacob, București	275
43.	Sorin Vultureanu, ***	268
44.	Adina Pampu, Făgăraș	265
45.	Bogdan-Gabriel Ionescu, Craiova	262
46.	Cosmin Molea, Bascov	260
47.	Marius-Dan Dragomir, Alba-Iulia	245
48.	Andrei-Gabriel Ion, Constanța	243
49.	Gabriel Kreindler, București	234
50.	Marius Nicolae, Slatina	233

Rundele #16 - #25

În continuare vă vom prezenta enunțurile celor 10 probleme propuse spre rezolvare la rundele 16-25 ale ediției din acest an a concursului de programare Bursele Agora.

P020424: Tăiere

Pe o tablă se află n cartonașe, așezate unul lângă altul pe o linie, de la stânga la dreapta. Pe fiecare dintre cartonașe este scrisă o cifră diferită de 0. Trebuie eliminate k grupuri a câte x cartonașe aflate pe poziții consecutive. În total vor fi eliminate $k \cdot x$ cartonașe. După eliminare, numărul obținut prin citirea cifrelor de pe cartonașe (de la stânga spre dreapta) trebuie să fie cât mai mic posibil.

Date de intrare

Prima linie a fișierului de intrare **CUT.IN** conține numărul n al cartonașelor, numărul k al grupurilor care trebuie eliminate și numărul x al cartonașelor dintr-un grup. Aceste numere vor fi separate prin spații. Cea de-a doua linie a fișierului conține cele n cifre scrise pe cartonașe, neseparate prin spații.

Date de ieșire

Fișierul de ieșire **CUT.OUT** va conține cel mai mic număr care poate fi obținut după eliminare.

Restricții

- $5 \le n \le 1000$;
- $1 \le k \le 100$;
- $1 \le x \le 9$;
- $k \cdot x < n$.

Exemplu

CUT.IN	CUT.OUT
7 2 3	2
5132789	

Timp maxim de execuție/test: 2 secunde

P020425: Sume

Pe o masă se află n cartonașe; pe fiecare dintre cartonașe este scris un număr natural. De pe masă pot fi alese oricâte cartonașe. După alegere se calculează suma numerelor de pe cartonașele alese. Va trebui să determinați numărul sumelor distincte care pot fi obținute prin astfel de alegeri.

Date de intrare

Prima linie a fișierului de intrare **SUM.IN** conține numărul n al cartonașelor. Cea de-a doua linie a fișierului conține cele n numere scrise pe cartonașe, separate prin spații.

Date de ieșire

Fișierul de ieșire **SUM.OUT** va conține numărul sumelor distincte care pot fi obținute.

Restricții și precizări

- $1 \le n \le 100$;
- numerele de pe cartonașe sunt cuprinse între 0 și 10000;
- pot fi alese între 1 și n cartonașe.

Exemplu

SUM.IN	SUM.OUT
5	11
0 1 2 3 4	

Timp maxim de execuție/test: 2 secunde

P020426: Warp

Imperiul Galactic a decis stabilirea unor rute galactice folosind găuri de vierme. Pentru a testa acest sistem au fost alese n planete care vor participa la testare. O gaură de vierme unește două planete distincte, iar în interiorul său se poate circula în ambele sensuri.

Pentru fiecare gaură de vierme se cunoaște cantitatea de energie *warp* (w) necesară pentru a o traversa. Dacă o navă intră în gaura de vierme având o cantitate de energie disponibilă e, atunci la ieșire va rămâne disponibilă o cantitate de energie e / w deoarece, indiferent de cantitatea inițială, prin trecerea prin gaura de vierme, energia se reduce de w ori. Se spune că este necesară o cantitate de energie w, deoarece o navă care ajunge să aibă o cantitate de energie mai mică decât 1 este distrusă instantaneu.

Folosind sistemul de găuri de vierme se poate ajunge de la o planetă la oricare alta. Totuși, în multe cazuri nu este posibilă călătoria directă între două planete. În aceste situații vor fi alese rute care conțin două sau mai multe găuri

de vierme. Va trebui să determinați cantitatea minimă de energie care este suficientă pentru a călători între oricare două planete.

Date de intrare

Prima linie a fișierului de intrare **WARP.IN** conține numărul n al planetelor, precum și numărul m al găurilor de vierme. Cele două numere sunt separate printr-un spațiu. Fiecare dintre următoarele m linii va conține câte trei numere întregi (x, y și w), separate prin spații. Aceste numere au semnificația: există o gaură de vierme între planetele identificate prin x și y, iar cantitatea de energie necesară supraviețuirii în interiorul găurii este w.

Date de ieșire

Fișierul de ieșire **WARP.OUT** va conține cantitatea minimă de energie care este suficientă pentru a călători între oricare două planete.

Restricții și precizări

- $2 \le n \le 100$;
- $1 \le m \le 1000$;
- cantitățile de energie, pentru a supraviețui în interiorul găurilor de vierme, sunt numere întregi cuprinse între 2 și 9;
- planetele sunt identificate prin numere cuprinse între 1 și n;
- între două planete poate exista cel mult o gaură de vierme.

Exemplu

WARP.IN

```
4 5
1 2 4
1 3 6
1 4 8
2 3 5
3 4 7
```

WARP.OUT

```
32
```

Timp maxim de execuție/test: 0,5 secunde

P020427: Găuri

Se consideră o matrice cu m linii și n coloane, ale cărei elemente pot avea fie valoarea 0, fie valoarea 1. O zonă este o mulțime de elemente care au valoarea 0.

Două elemente învecinate (pe verticală sau orizontală), care au ambele valoarea 0, vor face parte din aceeași zonă. O gaură este o zonă care nu conține elemente aflate pe prima linie, pe ultima linie, pe prima coloană sau pe ultima coloană a matricei. În elementele din găurile matricei trebuie scrise numere. Toate elementele din aceeași gaură trebuie să aibă aceeași valoare. Va trebuie să determinați cel mai mic număr care poate reprezenta suma elementelor din oricare dintre găuri.

Date de intrare

Fișierul de intrare **HOLES.IN** conține pe prima linie două numere naturale m și n, separate printr-un spațiu, care reprezintă dimensiunile matricei.

Pe fiecare dintre următoarele m linii se află un șir de n numere care pot fi 0 sau 1 și care nu sunt separate prin spații. Aceste numere reprezintă elementele matricei.

Date de ieșire

Fișierul de ieșire **HOLES.OUT** va conține cel mai mic număr care poate reprezenta suma elementelor din oricare dintre găurile matricei.

Restricții și precizări

- $3 \le m, n \le 100$;
- $1 \le M \le 10000$;
- va exista întotdeauna cel puțin o gaură.

Exemplu

HOLES.IN

```
10 10
1110010101
1010111111
1010110001
1111110011
1000010111
1011111111
1011111111
1000000000
1111111111
```

HOLES.OUT

```
6
```

Timp maxim de execuție/test: 0,5 secunde

P020428: Daneel

R. Daneel Olivaw trebuie să ajungă de pe *Trantor* pe *Terminus*. În acest scop el va trebui să folosească sistemul galactic de găuri de vierme. O gaură de vierme unește două planete distincte, iar în interiorul său se poate circula în ambele sensuri. Pentru fiecare gaură de vierme se cunoaște durata deplasării prin intermediul ei, precum și costul închirierii unei nave spațiale care trece prin acea gaură.

Daneel are la dispoziție o sumă limitată, dar dorește să ajungă cât mai repede pe *Terminus*. Așadar, el va trebui să facă o călătorie pe care și-o poate permite, și care să dureze cât mai puțin.

Date de intrare

Prima linie a fișierului de intrare **DANEEL.IN** conține numărul n al planetelor, precum și numărul m al găurilor de vierme. Cele două numere sunt separate printr-un spațiu. Cea de-a doua linie a fișierului va conține suma s pe care o are la dispoziție *Daneel*.

Fiecare dintre următoarele m linii va conține câte patru numere întregi (x, y, t și c), separate prin spații. Aceste numere au semnificația: există o gaură de vierme între planetele identificate prin x și y, durata călătoriei prin gaura de vierme este t, iar costul călătoriei este c.

Date de ieșire

Fișierul de ieșire **DANEEL.OUT** va conține durata minimă a călătoriei lui *Daneel*, în cazul în care aceasta poate avea loc. În caz contrar, fișierul va conține valoarea -1.

Restricții și precizări

- $2 \le n \le 100$;
- $1 \le m \le 10.000$;
- durata traversării unei găuri de vierme este un număr întreg cuprins între 1 și 100;
- costul traversării unei găuri de vierme este un număr întreg cuprins între 1 și 100;
- planetele sunt identificate prin numere cuprinse între 1 și n;
- *Trantor* este planeta identificată prin 1;
- *Terminus* este planeta identificată prin n;
- există întotdeauna posibilitatea de a ajunge de pe *Trantor* pe *Terminus*, deși este posibil ca *Daneel* să nu își poată permite o astfel de călătorie;
- între două planete pot exista mai multe găuri de vierme.

Exemplu

DANEEL.IN

```
4 5
15
1 2 2 3
1 3 1 6
1 4 15 1
2 3 3 1
3 4 5 10
```

DANEEL.OUT

```
10
```

Timp maxim de execuție/test: 3 secunde

P020429: Munte

Un număr are aspect de munte dacă există o poziție k, astfel încât $c_1 \le c_2 \le c_k$ și $c_k \ge c_{k+1} \ge ... \ge c_n$, unde c_1, ..., c_n sunt cifrele numărului.

Dându-se un număr, să se determine numărul minim de cifre care trebuie eliminate, astfel încât numărul rămas să aibă aspect de munte.

Date de intrare

Fișierul de intrare **MOUNTAIN.IN** conține o singură linie pe care se află numărul dat.

Date de ieșire

Fișierul de ieșire **MOUNTAIN.OUT** va conține o singură linie pe care se va afla numărul minim al cifrelor care trebuie eliminate.

Restricții și precizări

- numărul va conține cel mult 10000 de cifre;
- cifrele numărului sunt nenule.

Exemplu

MOUNTAIN.IN

```
248793
```

MOUNTAIN.OUT

```
1
```

Timp maxim de execuție/test: 2 secunde

P020430: Puncte

Se consideră n puncte în plan. Să se determine numărul posibilităților de a alege trei dintre aceste puncte, astfel încât aria triunghiului determinat de acestea să fie un număr întreg.

Date de intrare

Fișierul de intrare **POINTS.IN** conține pe prima linie numărul n al punctelor din plan. Fiecare dintre următoarele n linii va conține câte două numere, separate prin spații, reprezentând coordonatele unui punct.

Date de ieșire

Fișierul de ieșire **POINTS.OUT** va conține o singură linie pe care se va afla numărul posibilităților de a alege trei dintre puncte, astfel încât aria triunghiului determinat de acestea să fie un număr întreg.

Restricții și precizări

- $3 \le n \le 10.000$;
- coordonatele punctelor sunt numere întregi cuprinse între 0 și 1000;
- aria triunghiului determinat de trei puncte coliniare este considerată a fi 0;
- nu există două puncte aflate la aceleași coordonate.

Exemplu

POINTS.IN

```
0 0
0 2
2 2
2 0
```

POINTS.OUT

```
4
```

Timp maxim de execuție/test: 2 secunde

P020431: Era imperiilor

Împăratul francez dorește să-și împrejmuiască palatul. Muncitorii săi pot construi un număr de n ziduri. Acestea pot fi amplasate în orice direcție și în orice poziție, dar fiecare zid trebuie să aibă forma liniară.

El dorește ca palatul să fie împrejmuit complet, așadar sistemul de ziduri va avea forma unui poligon. Pentru fiecare zid se cunoaște lungimea acestuia. Va trebui să determinați aria maximă a regiunii care poate fi împrejmuită.

Date de intrare

Fișierul de intrare **AOE.IN** conține pe prima linie numărul n al zidurilor. Fiecare dintre următoarele n linii va conține câte un număr care reprezintă lungimea unui zid.

Date de ieșire

Fișierul de ieșire **AOE.OUT** va conține o singură linie pe care se va afla aria maximă a regiunii care poate fi împrejmuită de cele n ziduri.

Restricții

- $3 \leq n \leq 1000$;
- lungimile zidurilor sunt numere întregi cuprinse între 1 și 10000;
- aria maximă va fi scrisă cu o precizie de două zecimale exacte;
- lățimile zidurilor sunt neglijabile.

Exemplu

AOE.IN

```
4
1
1
1
1
```

AOE.OUT

```
1
```

Timp maxim de execuție/test: 2 secunde

P020432: Aladin

Aladin s-a gândit la un moment dat să intre în afaceri cu covoare, deoarece crede că acestea sunt foarte profitabile. El a reușit să rezolve toate problemele tehnice și acum poate să realizeze covoare de dimensiune $m \times n$.

Aceste covoare sunt împărțite în $m \times n$ celule de dimensiune 1×1. Fiecare celulă va fi colorată cu alb sau cu negru.

Datorită faptului că vrea să obțină covoare cu anumite proprietăți magice este necesar ca orice pătrat, de dimensiune 2×2, să conțină două celule colorate cu alb și două celule colorate cu negru.

Aladin dorește să știe câte covoare diferite, de dimensiune $m \times n$, care să beneficieze de proprietățile magice se pot fabrica.

Date de intrare

Fișierul de intrare **ALADDIN.IN** conține pe prima linie două numere întregi m și n, separate printr-un singur spațiu, care reprezintă dimensiunile pentru un covor.

Date de ieșire

Fișierul de ieșire **ALADDIN.OUT** trebuie să conțină o singură linie pe care se va afla un singur număr care reprezintă numărul de covoare de dimensiune $m \times n$ care se pot construi astfel încât acestea să beneficieze de proprietățile magice.

Restricție

- $2 \leq m, n \leq 10000$.

Exemplu

ALADDIN.IN	ALADDIN.OUT
4 2	18

Timp maxim de execuție/test: 1 secundă

P020433: Santinele

În drumul spre *Zion* nava *Nabucodonosor* este urmărită de santinele. Nava dispune de o armă pe bază de puls electromagnetic care, dacă este activată, atunci va distruge toate santinelele pe o rază de r metri în jurul ei.

Datorită faptului că nava este foarte avariată, arma nu mai poate fi activată decât o singură dată.

Știind că în drumul spre *Zion* santinelele nu își schimbă formația, că nava se poate deplasa oriunde între santinele și că nava și santinelele se află tot timpul în același plan se cere să se determine numărul maxim de santinele care pot fi distruse prin activarea armei cu puls electromagnetic.

Date de intrare

Fișierul de intrare **SENTINEL.IN** conține pe prima linie numărul n al santinelelor și raza r a armei cu puls electromagnetic, separate printr-un singur spațiu.

Fiecare dintre următoarele n linii va conține câte două numere, separate prin spații, reprezentând coordonatele la care se află o santinelă relativ la un observator care se deplasează cu aceeași viteză cu cea a santinelelor.

Date de ieșire

Fișierul de ieșire **SENTINEL.OUT** trebuie să conțină o singură linie pe care se va afla un singur număr care reprezintă numărul maxim de santinele care pot fi distruse prin activarea armei cu puls electromagnetic.

Restricții și precizări

- $1 \leq n \leq 1000$;
- raza r a armei cu puls electromagnetic este un număr întreg cuprins între 1 și 1000;
- coordonatele santinelelor sunt numere întregi cuprinse între 0 și 1000;
- nu există două santinele la aceleași coordonate.

Exemplu

SENTINEL.IN

```
3 2
2 2
0 2
2 0
```

SENTINEL.OUT

```
2
```

Timp maxim de execuție/test: 1 secundă

$X \wedge Y \wedge X = Y...$

To be XOR not to be

Lucian Ilea, Bogdan Mărincaș

În cadrul acestui articol vă vom prezenta câteva proprietăți ale operatorului XOR și câteva aplicații în cadrul cărora acesta poate fi folosit cu succes.

În cadrul anumitor limbaje de programare, pe lângă operatorii logici binecunoscuți **or**, **and**, **not** se poate folosi și operatorul "*sau exclusiv*", exprimat prin cuvântul rezervat **xor** în *Pascal* și prin caracterul "^" în *C/C++*.

În afara simplificării scrierii unor expresii logice, acest operator logic poate fi folosit și pentru calcule matematice, în cazurile în care operanzii sunt numere întregi fără semn.

Dacă x și y sunt două variabile logice, atunci $x \wedge y$ are valoarea 1 dacă și numai dacă exact unul dintre operanzi are valoarea 1. Cu alte cuvinte $x \wedge y = (!x \,\&\, y) \,||\, (x \,\&\, !y)$. Operatorul *binar* ^ are importante proprietăți algebrice ușor de verificat: asociativitate, comutativitate, 0 este elementul neutru al operației, $x \wedge x = 0$ și $x \wedge !x = 1$.

Prezentăm acum câteva aplicații ale operatorului **xor**.

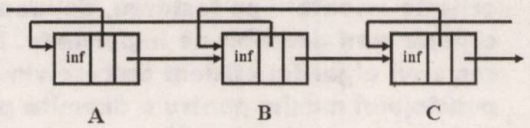
1. Prima aplicație pe care o prezentăm este cea de interschimbare a valorilor a două numere întregi fără semn x și y. Această operație o putem realiza pe baza "*regulii celor trei pahare*" ($aux = x$; $x = y$; $y = aux$) sau folosind adunarea și scăderea ($x = x + y$; $y = x - y$; $x = x - y$). Este oarecum surprinzător că poate fi folosită și secvența de instrucțiuni $x = x \wedge y$; $y = x \wedge y$; $x = x \wedge y$, incredibil de concisă.

2. Fie următoarea problemă: *Se consideră un șir de numere în care, cu excepția uneia, fiecare valoare întâlnită apare de un număr par de ori. Se cere determinarea singurului element care apare de un număr impar de ori.*

De exemplu, pentru $n = 7$ și elementele șirului x egale cu 4, 3, 5, 3, 4, 4, 4 rezultatul este 5. În rezolvare vom aplica proprietățile operatorului ^. Comutativitatea și asociativitatea ne oferă dreptul de a efectua calculele în orice ordine. Faptul că $x \wedge x = 0$ ne asigură că dublurile din șir se anulează reciproc și că, în final, efectuând disjuncție exclusivă între toate elementele șirului x, va rămâne doar elementul cu număr impar de apariții. (Se observă că nu este necesar să se folosească șiruri.)

3. În aceeași categorie putem include și următoarea problemă: *Se consideră un număr natural n și $n - 1$ numere naturale distincte aparținând intervalului* $[1, n]$. *Să se determine singurul număr natural aparținând intervalului care nu apare în acest șir.* Răspunsul îl considerăm ușor de obținut.

4. Următoarea aplicație interesantă o putem realiza pe liste dublu înlănțuite. Listele dublu înlănțuite, de regulă, conțin un câmp pentru informații, un câmp de legătură spre elementul anterior și un câmp de legătură spre elementul următor. Are loc o înlănțuire ca în figura următoare.



În cazul *listelor dublu înlănțuite XOR* informațiile de legătură sunt comasate într-un singur câmp. De exemplu, pentru elementul A aceasta va fi $(A - 1) \wedge B$, pentru B va fi $A \wedge C$, etc. (se efectuează disjuncție logică între adresele elementelor vecine). Când se traversează lista de la stânga la dreapta se va lua adresa elementului anterior și se va efectua ^ cu valoarea din câmpul de legătură. Similar se va proceda pentru traversarea de la dreapta la stânga. Apare dezavantajul că operațiile de inserare și de ștergere de elemente presupun calcule suplimentare. Deoarece determinarea unei noi adrese se face în funcție de două adrese anterioare, cea mai simplă implementare va folosi două santinele.

Dintre aplicațiile mai importante ale disjuncției exclusive menționăm: *viruși informatici*, *realizarea animației* și *securitatea datelor*. Unii dintre virușii informatici "infectează" datele dintr-un calculator aplicându-le un șablon ("semnătură") cu care efectuează operația ^ la nivel de bit. Este inutil să precizăm că "devirusarea" constă în aplicarea încă o dată a aceleiași matrițe.

Menționăm că unele probleme în care diverse variabile pot avea doar două stări (0 sau 1) care se pot modifica alternativ sunt de fapt probleme de aplicare a operatorului ^. A se vedea în acest sens problema "*XOR*" de la *IOI* 2002, *Coreea de Sud* (*GInfo* nr. 12/8). Însă, în cazul acestei probleme principala dificultate este de altă natură...

Dl. prof. Lucian Ilea este cadru didactic și Bogdan Marincaș este elev în clasa a XII-a la Colegiul Național "Emil Racoviță" din Cluj-Napoca. Pot fi contactați prin e-mail la adresa **ilucian@email.ro**, *respectiv* **bond007@xnet.ro**.

Sfaturi pentru viitori olimpici (1)

TESTAREA și DEPANAREA programelor

Mihai Stroe

Datorită apropierii fazelor avansate ale olimpiadelor, inițiem o serie de două articole orientate pe testarea, depanarea și optimizarea programelor. Aceste aspecte sunt deosebit de importante. În concursurile de informatică, elaborarea unui algoritm eficient de rezolvare nu este suficientă pentru obținerea punctajului maxim pentru o anumită problemă; trebuie scris un program, care trebuie testat, depanat și eventual optimizat.

Articolul curent tratează problema testării și depanării programelor; următorul număr al *GInfo* va conține un articol care se ocupă de optimizarea acestora...

Cadrul general

Mulți concurenți "economisesc" timp de implementare, scriind programe greu de urmărit. De multe ori, mult mai mult timp se pierde depanând astfel de programe; acest timp ar fi putut fi economisit programând atent și îngrijit.

În continuare, vom presupune că vă aflați în următoarea situație:

- aveți o idee de rezolvare pentru o anumită problemă;
- ați verificat că algoritmul găsit funcționează pe toate cazurile și sunt șanse mari ca o implementare eficientă să se încadreze în timpul de execuție;
- trebuie să scrieți un program care să rezolve problema.

Scrierea unui program ușor de urmărit și de depanat

Claritatea și structurarea codului

Procesul implementării constă în a scrie și a depana simultan programul. Cu cât greșelile sunt depistate mai rapid, cu atât este mai bine.

Pentru a fi ușor de urmărit și ușor de depanat, un program trebuie să fie structurat. Astfel, diferitele etape din cadrul rezolvării trebuie să fie conținute în funcții și proceduri diferite sau, dacă sunt conținute în același subprogram din motive de eficiență, vor fi separate prin spații libere și comentarii.

Numele subprogramelor trebuie să fie explicative (de exemplu "init" pentru o procedură de inițializare, "pantă" pentru un subprogram care determină panta unei drepte etc.), astel sugerând și "funcția" pe care o îndeplinesc.

Încapsularea etapelor în subprograme diferite ușurează și testarea programului.

Imediat după ce a fost scris, fiecare subprogram sau program poate fi testat (și depanat, dacă este cazul). Este mult mai ușor să detectăm o eventuală eroare dacă examinăm o secvență de cod relativ scurtă, decât dacă examinăm întreg programul. Fiecare subprogram conține rezolvarea unei subprobleme care la rândul ei are date de intrare și date de ieșire (rezultate) care astfel pot fi verificate imediat.

Alternativa, uneori mai eficientă din punct de vedere al timpului de implementare (și recomandată pentru avansați), constă în a presupune că toate subprogramele vor fi scrise corect; chiar și în acest caz, delimitarea subprogramelor simplifică testarea și depanarea. Dacă se detectează o eroare, subprogramele pot fi testate unul câte unul.

Un exemplu de testare incrementală constă în testarea imediată a procedurii de citire a datelor și a celor care decurg imediat din ea, cum ar fi inițializarea unor structuri. Aceasta necesită scrierea unor teste (le veți scrie oricum, pentru a testa programul; de ce să nu le scrieți de la început?) și executarea procedurii de citire imediat după scrierea ei. Această testare se recomandă atunci când formatul datelor de intrare este complex. De asemenea, erorile de intro-

ducere a datelor de test se detectează mult mai ușor în această fază.

Ați scris un program ușor de urmărit dacă, uitându-vă din nou la el peste șase luni, veți înțelege ce și cum face și îl veți putea urmări pas cu pas. Pentru aceasta, următoarele indicații pot fi folositoare:

- organizați programul astfel încât să reflecte ideea de rezolvare;
- folosiți facilitățile de indentare oferite de mediul de dezvoltare;
- folosiți spații libere între blocuri de cod care realizează acțiuni diferite;
- folosiți comentarii; cele câteva secunde "pierdute" pentru a scrie un comentariu explicativ pot reduce cu câteva minute timpul de depanare;
- folosiți nume de variabile sugestive (de exemplu, un program care conține variabile `a`, `aa` și `aaa` poate deveni greu de urmărit) și încercați, pe cât posibil, să nu folosiți aceeași variabilă în scopuri diferite (excepțiile se admit dacă refolosirea este necesară datorită limitărilor de memorie);
- expresiile aritmetice care apar trebuie să fie ușor de urmărit; astfel, se recomandă folosirea parantezelor pentru a clarifica ordinea operațiilor și folosirea spațiilor libere în expresii etc.;
- încercați să folosiți cât mai puțin metoda "copy-paste și modificări minore" pentru a rezolva subprobleme similare, deoarece puteți greși la acele modificări; pe deasupra, lungimea programului rezultat va fi mai mare, deci depanarea va fi mai dificilă; dacă eficiența nu este afectată, se recomandă scrierea unor funcții/proceduri care vor fi apelate cu parametri diferiți.

Comentariile trebuie să fie scurte (nu scrieți "povești"). Ele se folosesc pentru a delimita și a explica rezolvarea anumitor subprobleme (mai ales în cazul în care, din motive de eficiență, acestea sunt tratate în același subprogram) și pentru a explica aspecte de implementare. De exemplu, o expresie aritmetică poate arăta "ciudat" după câteva optimizări; în acest caz, ea va fi precedată, într-un comentariu, de expresia "ușor de citit" și, eventual, de semnificația ei.

Se folosesc comentarii și pentru a explica semnificația și modul de calcul al elementelor unor anumite matrice. De exemplu, fie o problemă de *programare dinamică*, în care N persoane se împart în K grupe, în fiecare grupă având persoane consecutivi din șir, astfel încât să se obțină o penalizare minimă, după anumite criterii. Valoarea $A[i, j]$ poate reprezenta "penalizarea minimă care se obține împărțind primii j oameni în i grupe". Ar fi neplăcut să implementați rezolvarea, apoi să recitiți enunțul, să vă mutați atenția asupra unor alte aspecte și... să afișați $A[N, K]$ în loc de $A[K, N]$, deoarece ați uitat ce semnificau indicii. Un comentariu bine plasat micșorează posibilitatea apariției unei astfel de erori.

De asemenea, pe parcursul depanării, se scrie câte un comentariu pentru fiecare secvență al cărei conținut nu este clar la prima recitire.

Detectarea existenței erorilor

Evident, dacă programul se termină printr-un *crash*, sau rezultatul afișat de program pe un anumit test nu corespunde așteptărilor, sunt mari șanse ca în program să existe o eroare. Verificați că ați înțeles bine problema și că ați tastat corect valorile din fișierul de intrare!

O mare parte din erori sunt detectate și localizate folosind opțiunile de compilare. În *Pascal*, acest lucru se face folosind *directivele de compilare*; cele mai importante sunt `$R` - *range checking*, `$S` - *stack checking* și `$I` - *input/output checking*. Prezența unei anumite directive de compilare în starea "`+`" instruiește compilatorul să genereze cod care să facă anumite verificări; de exemplu, la `$R+` se verifică domeniul variabilelor și al indicilor, la `$S+` depășirea stivei, iar la `$I+` succesul operațiilor de intrare/ieșire.

Prezența fragmentelor de cod suplimentare duce la detectarea mai ușoară a unor erori "clasice". Pe de altă parte, executarea lor poate încetini considerabil programul. De aceea, se recomandă ca directivele de compilare să fie setate pe "`+`" la testarea pentru detectarea erorilor și pe "`-`" la testarea pentru eficiență, folosind teste mari. În programul final, directivele vor fi setate pe "`-`".

Multe erori pot fi detectate și localizate mai ușor dacă programul face verificări pe parcurs și afișează informații importante în timpul execuției.

Pentru anumite părți ale programului este util să scrieți subprograme care să ușureze depanarea, verificând starea anumitor variabile și structuri de date. De exemplu, dacă sunteți începător în lucrul cu *heap*-uri, puteți scrie un subprogram care să testeze dacă *heap*-ul pe care îl folosiți este valid. În etapa de depanare veți introduce câteva apeluri ale acestui subprogram; dacă la un moment dat *heap*-ul nu este valid, ați detectat o eroare.

O altă metodă de a ușura depanarea este să afișați pe ecran valorile anumitor variabile, în locurile esențiale. Dacă, la execuția pe un test simplu, o parte din variabile nu au valorile așteptate, au fost detectate erori. Un efect similar se poate obține folosind *watches* și *breakpoint*-uri, dar în prima fază poate fi mai eficient să afișăm pe ecran și să examinăm valorile la sfârșit. Dacă programul este corect, nu mai este nevoie de *watches* și *breakpoint*-uri, iar procesul de testare durează mult mai puțin.

Metoda de mai sus nu pare aplicabilă dacă ați dori să afișați mai mult de 25 de valori pe linii diferite. Pentru un astfel de caz, puteți face afișarea într-un fișier; astfel, programul va avea un fișier de ieșire și un fișier "`log` al execuției". Puteți examina ambele fișiere după rulare.

Liniile și subprogramele care ajută la detectarea erorilor nu vor fi șterse după testarea și depanarea programului.

Alternativa, mult mai utilă, constă în a le transforma în comentarii. Astfel, ele pot fi refolosite; dacă programul este corect, dar nu și suficient de rapid, rescrierea și optimizarea unor subprograme poate duce la apariția unor noi erori.

O întrebare frecventă este: *Dar oare toate aceste precauții, scrierea comentariilor etc. nu necesită prea mult timp?* Răspunsul este că timpul pierdut cu scrierea unui program ușor de depanat este câștigat în faza de depanare. În plus, o parte din etapele prezentate, cum ar fi cea în care scriem subprograme pentru a verifica starea unor structuri de date, pot fi amânate pentru începutul etapei de depanare, și folosite în cazul apariției unor erori. Este mult mai ușor să scriem o astfel de funcție, decât să examinăm de fiecare dată structura "ochiometric". Aveți grijă să nu greșiți tocmai subprogramul de verificare!

Localizarea și corectarea erorilor

Ați scris un program, dar acesta nu dă rezultatul dorit. Mai mult, știți că de la un anumit moment încolo programul nu se comportă corect. Poate a afișat pe ecran o valoare necorespunzătoare, sau poate s-a oprit cu o eroare de depășire a stivei. Ce faceți în continuare?

Erori frecvente

În această subsecțiune vom enumera cele mai frecvente erori. Unele dintre ele pot fi evitate în faza de scriere a programului. În orice caz, dacă programul nu se comportă cum doriți, există mari șanse ca motivul să fie unul din cele de mai jos. Acestea sunt deci principalele erori pe care trebuie să le aveți în vedere pe parcursul depanării.

Folosirea pointerilor poate introduce erori greu de detectat. De asemenea, multe din structurile alocate dinamic (de exemplu, listele) sunt dificil de depanat (nu putem vedea o listă în fereastra de *watches*). De aceea, se recomandă evitarea alocării dinamice; dacă totuși nu este suficientă memorie pentru a aloca totul static, puteți aloca dinamic vectori, matrice etc.

În cazul în care aveți o matrice prea mare (de exemplu o matrice de numere întregi A cu 300 de linii și 300 de coloane, care nu încape în 64K) veți aloca liniile acesteia ca vectori. Veți avea alocat static un vector de pointeri către liniile matricei; în *Pascal*, un element al matricei va fi accesat cu `A[i]^[j]`.

Dacă folosiți o structură de pointeri, este bine să scrieți subprograme care vă afișează pe ecran conținutul structurii într-o formă inteligibilă, pentru a facilita depanarea. De exemplu, se poate vizualiza valoarea atașată elementului curent în prelucrare `p^.info`.

Un alt tip frecvent de erori este cel al depășirii limitelor tipurilor de date. De exemplu, în *Pascal*, adunând două valori care depășesc *MAXLONGINT*, putem obține un rezultat negativ. Este datoria voastră să vă protejați față de astfel de depășiri.

Neinițializarea variabilelor poate conduce la erori dificil de detectat. De exemplu, la alocarea dinamică a unui vector, acestuia i se rezervă o zonă de memorie, dar ea nu este inițializată cu 0. Un program care presupune că zona este inițializată cu 0 se va comporta într-o manieră imprevizibilă. Dar nici variabilele locale ale unor subprograme nu se inițializează cu zero.

Calculele cu numere reale pot introduce erori. De exemplu, testarea egalității a două numere reale nu se face cu `"x = y"`, ci cu `"abs(x - y) < eps"`, unde `eps` este o constantă cu o valoare foarte mică, de exemplu `0.0000001`.

Trecerea bruscă a unui programator de la *Pascal* la *C* poate fi cauza unor erori clasice. De exemplu, programatorul *Pascal* este obișnuit cu **`array`**-uri care se indexează începând de la 1; în *C*, un **`array`** cu N elemente are indicii elementelor de la 0 la N - 1. De asemenea, ciclurile **`for`** se scriu `"for(i = 0; i < N; i++);"`, iar fostul programator *Pascal* este tentat să scrie `"i <= N"`. Același programator poate folosi `"="` pentru comparare în loc de `"=="`. Recomand ca trecerea de la *Pascal* la *C* să se facă în timp util, astfel încât programatorul să se poată obișnui cu noul limbaj (deci cu cel puțin 2-3 luni înainte de concursurile importante).

O altă eroare constă în nedeclararea unor variabile locale acolo unde este nevoie. De exemplu, o procedură recursivă care folosește o variabilă contor **`i`** trebuie să o declare local; folosirea variabilei globale nu asigură restaurarea ei automată după fiecare întoarcere din apelurile efectuate. Pe de altă parte, această variabilă ocupă spațiu în stivă; din motive de eficiență, se poate prefera folosirea unei variabile globale și scrierea unor instrucțiuni în scopul restaurării ei.

Ultimul tip de erori din listă, dar poate cel mai des întâlnit, este cel al erorilor "de neatenție". Acestea sunt, probabil, erorile care au costat participanții la olimpiade mai multe puncte (ușor de obținut) decât orice altceva în ultimii 10 ani. Sunt motivele pentru care unul dintre cei mai valoroși concurenți poate pierde o medalie sau o calificare la un concurs internațional.

Astfel de erori sunt: scrierea unui indice **`i`** în loc de un **`j`** într-un ciclu **`for`**, afișarea valorii $A[n, k]$ în loc de $A[k, n]$, scrierea unui subprogram (de exemplu, subprogramul de inițializare) și neapelarea lui când este nevoie, intershimbarea numărului de linii n cu numărul de coloane m etc. Dacă obișnuiți să faceți astfel de erori, este foarte important să citiți cu atenție secțiunile programului pe care vă concentrați în timpul depanării. O măsură preventivă constă în a reciti fiecare subprogram imediat după ce l-ați scris; șansele de a observa dacă aveți o astfel de eroare sunt mult mai mari dacă examinați o secțiune mică de cod.

Tot în categoria erorilor de neatenție încadrăm netratarea anumitor cazuri particulare care pot apărea (de exemplu, graful cu 0 noduri, dacă nu se specifică explicit în enunț că nu putem întâlni un astfel de caz), nerespectarea formatului de afișare, afișarea în alt fișier decât cel specifi-

cat în enunț și nerezolvarea unui subpunct al problemei pentru că nu ați citit cu atenție enunțul. Fiți foarte atenți cu aceste tipuri de erori, și citiți enunțul și formatul de afișare de cel putin 3-4 ori! Mulți programatori experimentați, printre care și autorul acestui articol, au pierdut sute, poate mii de puncte din cauza unor astfel de erori.

Este recomandat să cunoașteți care sunt erorile pe care le faceți cele mai frecvent. Gândiți-vă la modalități pentru a le evita (de exemplu, dacă aveți erori datorate lucrului cu pointerii, încercați să alocați totul static!) și pentru a le descoperi rapid în cazul în care totuși apar. Dacă ajungeți la observații utile și cu aplicabilitate generală, le puteți trimite redacției, și poate le veți regăsi în următorul articol pe această temă!

Depanarea propriu-zisă

Fără a insista prea mult asupra celor prezentate, reamintim că anumite operațiuni care nu au fost făcute anterior (cum ar fi afișarea valorilor unor variabile pe ecran) pot fi introduse în această fază.

În continuare ne vom concentra asupra depanării folosind *watches* și *breakpoint*-uri.

Elevii începători (și nu numai) ar trebui să învețe să stăpânească foarte bine instrumentele de depanare puse la dispoziție de mediul de programare. De asemenea, se recomandă învățarea *shortcut*-urilor prin care acestea pot fi folosite. Timpul de depanare poate fi redus semnificativ dacă, pe lângă scrierea unui program structurat, *breakpoint*-urile sunt plasate în punctele esențiale, se urmărește execuția în interiorul subprogramelor numai dacă este cazul etc.

Există un punct până în care programul funcționează corect și un punct în care valorile unor variabile nu sunt cele așteptate. Inițial, primul punct este începutul programului, iar ultimul este sfârșitul sau între aceste două puncte se află cel puțin o eroare. O parte din strategiile descrise mai sus pot restrânge acest interval.

Vrem să restrângem intervalul până când localizăm eroarea (sau una dintre ele) într-o secvență mică de cod, după care o putem observa și elimina.

Primul pas constă în punerea în *watches* a variabilelor care vor fi urmărite. Puneți variabilele care pot fi afectate și cele care vă ajută să înțelegeți ce se întâmplă. De exemplu, dacă aveți o problemă cu N puncte în plan, coordonatele punctelor vor fi în *watches* în majoritatea cazurilor, chiar dacă ați stabilit că sunt citite corect și nu sunt modificate ulterior.

Găsiți subprogramul în care este localizată eroarea, folosind *breakpoint*-uri și opțiunea *Step Over* din meniul *Debug* (*shortcut*-ul în *Borland Pascal* și *Borland C* este *F8*). Puneți *breakpoint*-uri între punctul de început (până la care totul este O.K.) și punctul de sfârșit, apoi actualizați-le în timp ce executați bucăți din program, într-o manieră similară căutării binare. Reluați execuția și restrângeți cât mai mult intervalul.

Dacă pe parcursul acestei operații vă gândiți că o anumită secvență de cod poate fi "suspectă", examinați-o pe aceea înainte de a continua procesul descris anterior. Puneți un *breakpoint* înainte și unul după, apoi rulați programul pentru a vă confirma sau infirma suspiciunea.

Un instrument de depanare foarte util, dar neglijat de unii, sunt *breakpoint*-urile condiționale. Concret, uneori se poate ca eroarea să nu apară la prima trecere printr-un anumit punct, ci după mai multe treceri. Setarea unui *breakpoint* și execuția (cu *Ctrl+F9* în *Borland*) până la detectarea erorii poate necesita timp. Soluția constă în folosirea unui *breakpoint* condițional. Un astfel de *breakpoint* se activează (duce la întreruperea execuției) când o anumită condiție este îndeplinită.

O metodă eficientă de a folosi *breakpoint*-urile condiționale este bazată pe *căutarea binară*. Contorizați numărul de treceri printr-o anumită zonă (declarați o variabilă contor pe care o incrementați la fiecare trecere) și verificați, în condiția *breakpoint*-ului, dacă aceasta a ajuns la o anumită valoare. Dacă eroarea încă nu a apărut, creșteți valoarea din condiție; dacă a apărut, o scădeți etc.

În cazurile în care eroarea este semnalată printr-o anumită combinație (pe care o cunoașteți - de exemplu, o distanță negativă) de valori ale variabilelor din *watches*, iar combinația este ușor de detectat, folosiți un *breakpoint* condițional și opriți execuția exact în locul unde ați detectat eroarea. Dacă este mai dificil, dar nu imposibil să detectați combinația care conduce la apariția erorii, scrieți o funcție care verifică, la fiecare trecere, dacă eroarea a apărut. În momentul în care apare eroarea, întrerupeți execuția (folosind un *breakpoint*) și examinați contextul.

Când aveți o eroare într-un program, este recomandat să nu considerați că o anumită secțiune este 100% corectă. Autorul, după ani de experiență, a reușit "performanța" de a face greșeli de neatenție în cadrul unor subprograme clasice, cum ar fi unul care implementa *algoritmul Roy-Floyd* pentru determinarea distanțelor minime între oricare două noduri dintr-un graf, și de a depana inutil, timp de mai multe minute, alte părți mult mai "suspecte" ale programului până la localizarea erorii. În acest caz, principiul "nevinovației până la proba contrarie" nu se aplică; toate subprogramele pot conține erori.

Rescrierea

V-ați aflat vreodată în situația de a depana pentru un timp îndelungat o secțiune dintr-un program, în care sunteți sigur că există o eroare, fără a reuși să o detectați?

O posibilă soluție constă în rescrierea secțiunii. Rescrieți secțiunea în alt mod, cât mai clar posibil, aplicând observațiile din acest articol. Nu optimizați decât după ce ați eliminat eroarea. Cu puțin noroc, s-ar putea ca rescrierea să nu mai conțină erori, sau acestea să fie mai ușor de depanat.

În orice caz, păstrați și versiunea anterioară (în comentarii sau în alt fișier). S-ar putea să refolosiți anumite bucăți de cod (să sperăm că nu și cea cu eroare...).

Generarea datelor de test

Depanarea nu se poate face în absența datelor de test. Acest pas este de fapt unul dintre primii pași ai rezolvării problemei; se recomandă ca algoritmul să fie verificat pe testul din exemplu și pe alte câteva teste, înainte de a trece la implementare.

Se știe că, pentru a crește probabilitatea de detectare a erorilor, datele de test trebuie să fie acoperitoare (cazuri particulare, cazuri de teste mari etc.) și variate. Totuși, în multe cazuri, elevii folosesc testul din exemplul problemei (care poate fi total neconcludent), mai scriu maxim două teste mici "de mână" și apoi se declară satisfăcuți și consideră că au rezolvat problema. Mulți dintre acești elevi obțin punctaje mici, după care unii dintre ei depun contestații, susținând că programul lor "mergea pe exemplu". Intenția noastră nu este de a-i ridiculiza pe acești elevi, ci de a-i ajuta să obțină punctele pe care le merită și de care, probabil, îi desparte o eroare pe care ar putea-o detecta și corecta în câteva minute.

În general, testele pe care un elev și le dă în timpul concursului se împart în două categorii:

- *teste pentru depanare rapidă* - aceste teste sunt în general de dimensiuni mici. Elevul poate urmări ușor comportarea programului, poate sesiza dacă valorile unor variabile nu sunt cele așteptate etc.; în această categorie intră testul din exemplu și încă 2-3 teste mici generate manual;
- *teste pentru eficiență* - acestea sunt în general testele mari. Dacă intrarea constă într-un număr, testele mari se generează manual. Pe de altă parte, dacă datele de intrare au o structură mai complexă (de exemplu, un graf) este util să se scrie un mic program care generează date de test. Se pot genera date de test aleatoare și cazuri limită (de exemplu, o matrice 100×100 plină cu 0). Cu ajutorul testelor din această categorie se verifică dacă programul are un comportament normal pentru seturi mari de date (de exemplu, nu se blochează), dacă se încadrează în timpul de execuție și dacă rezultatul afișat "pare" corect. Generatorul poate fi apoi folosit pentru a construi teste aleatoare, de dimensiuni mici și medii, pentru care verificarea corectitudinii rezultatului afișat este mai simplă; acestea pot fi folosite pentru a detecta erori care au "scăpat" de faza testelor mici generate manual.

În unele cazuri se recomandă scrierea unui program pentru verificarea unor caracteristici ale datelor de ieșire afișate. De exemplu, dacă soluția constă într-un drum de lungime minimă într-un graf, se poate scrie un program care verifică dacă muchiile afișate aparțin într-adevar grafului, dacă lungimea drumului este cea scrisă în fișier etc. Acest program folosește o parte din codul sursă al rezolvării problemei, cum ar fi subprogramul de citire a datelor.

Dacă scrieți astfel de programe pentru generare de teste și verificare, aveți grijă să nu greșiți la conceperea sau la scrierea lor! De exemplu, dacă intrarea problemei constă într-un graf conex, dar generatorul construiește grafuri nu neaparat conexe, s-ar putea să căutați o greșeală inexistentă în rezolvarea problemei!

Pe parcursul testării, citiți datele din fișierul de intrare cu numele din enunț și respectați formatul datelor de intrare! Ar fi neplăcut să pierdeți punctele acordate pentru o problemă pentru că faceți citirea din alt fișier sau scrieți rezultatele pe ecran.

În afară de testarea întregului program, poate fi util să testați și să depanați, independent, anumite subprograme. De exemplu, fie o procedură care calculează înfășurătoarea unei mulțimi de puncte. Poate că folosiți această procedură într-un context care nu se deduce direct din datele de intrare ale problemei. Pentru a o testa, puteți scrie o procedură auxiliară care citește dintr-un fișier coordonatele a N puncte, apoi scrieți/generați un astfel de fișier, apelați doar procedurile de citire și calcul al înfășurătorii convexe și examinați rezultatele.

După ce ați terminat testarea, depanarea și optimizarea, asigurați-vă că programul respectă formatul de afișare din enunț. Transformați în comentarii instrucțiunile care produceau *output* suplimentar pentru depanare și instrucțiunile care așteptau ca utilizatorul să apese o tastă, dacă aveți așa ceva (de exemplu, dacă ați afișat ceva pe ecran). La evaluare, nimeni nu va apăsa o tastă după terminarea programului (în afară de cazul în care acest lucru este specificat explicit în enunț), ceea ce va duce la depășirea timpului de execuție!

Concluzii

Testarea și depanarea programelor sunt operațiile care vă asigură că algoritmul de rezolvare de care sunteți atât de încântat va fi concretizat într-un program care va obține punctajul dorit.

Pentru începători, s-ar putea ca unele dintre indicațiile din acest articol să nu fie suficient de clare, deoarece se pot referi la situații pe care aceștia nu le-au întâlnit încă. Recitiți acest articol din când în când; privindu-l prin prisma experienței câștigate, s-ar putea să obțineți noi informații utile.

Este foarte important să tratați cu seriozitate etapele de testare și depanare. Pe măsură ce le veți stăpâni din ce în ce mai bine și anumite activități vă vor intra în reflex, timpul suplimentar alocat lor va scădea, ca și numărul de erori cu care vă veți confrunta.

Mențiuni

Țin să precizez faptul că acest articol este inspirat din articolele având aceeași temă de pe site-ul web al *USA Computing* (**http://ace.delos.com/usacogate**), site care este dedicat pregătirii elevilor și studenților din toată lumea pentru adevăratele concursuri de informatică.

Mihai Stroe este masterand la Universitatea Politehnica din București și poate fi contactat prin e-mail la adresa **mihai_stroe@yahoo.com.**

Mai multe despre PARTIȚII

Claudiu Soroiu

Ce reprezintă partiționarea unui disc? În cadrul acestui articol voi răspunde la această întrebare și voi descrie structura de date care se folosește pentru a reține informațiile despre partițiile unui disc, unde este localizată pe disc această structură și modurile în care se poate accesa un disc.

Introducere

Partiționarea reprezintă împărțirea unui disc fizic în mai multe unități logice. Nevoia de partiționare a apărut în momentul în care, datorită utilizării defectuoase a sistemelor de calcul, au început să se piardă date. Un alt motiv care a a contribuit la nevoia de partiționare a discurilor fizice a fost cel al instalării mai multor sisteme de operare pe același disc.

Partițiile se folosesc, de obicei, pentru a delimita fișierele utilizate de sistemul de operare de cele ale utilizatorilor. Utilizarea partițiilor duce la diminuarea riscului de a pierde date, deoarece, în cazul unei utilizări defectuoase a sistemului de operare se vor pierde fișiere sau date localizate în fișiere ale sistemului de operare care poate fi, în cel mai defavorabil caz, reinstalat.

Conceptul de partiție a unui disc s-a aplicat doar pentru *hard-disk*-uri, acestea fiind cele mai utilizate medii de stocare a informației și care au ajuns să aibă dimensiuni destul de mari. Nu se poate vorbi de partiționarea altor medii de stocare deoarece au dimensiuni mai mici sau în cazul în care au dimensiuni mari sunt folosite pentru a crea copii de siguranță, astfel permițând recuperarea informațiilor în cazul pierderii lor.

După cum se observă în figura 1, am împărțit un hard-disk în trei partiții:

- *S.O.* – această partiție va fi folosită de către sistemul de operare și va conține fișierele acestuia și alte aplicații necesare operării pe calculator;
- *Date de lucru* – această partiție va fi folosită de către utilizator pentru a stoca datele utilizate, date care cuprind documente și aplicații realizate de către utilizator;
- *Backup* – această partiție va fi utilizată pentru a salva copii de siguranță pentru anumite documente.

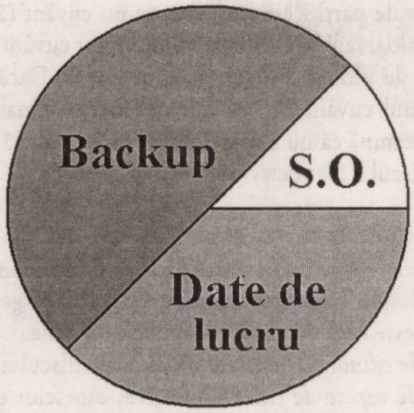


Figura 1: Hard-disk partiționat

De remarcat faptul că, în cazul utilizării defectuoase a a unui sistem de calcul care are discul partiționat după cum indică figura de mai sus, riscul apariției unei erori în cadrul celei de-a treia partiții (*Backup*) este aproape 0.

Primul sector

V-ați întrebat vreodată ce însemnă "*Track 0 bad or disk unusable*"? Acesta este un mesaj specific aplicației `format`, care se folosește pentru a pregăti un disc în vederea utilizării lui. Acest mesaj ne spune că discul nu mai poate fi utilizat. *De ce*? Răspunsul la această întrebare este cât se poate de simplu. *Deoarece prima pistă a discului, și mai exact primul sector al acestuia, conține o eroare de alt tip decât cel logic.*

Primul sector al unui *hard-disk* conține informații primare despre felul cum este partiționat discul și mai conține un program care este încărcat de *BIOS*-ul sistemului de calcul după ce acesta a fost pornit.

Primul sector poartă denumirea de *MBR* (*Master Boot Record* - informații primare pentru pornire).

Vom considera în continuare că un sector nu are o dimensiune mai mare de 512 bytes. În caz că se utilizează un disc ale cărui sectoare au o dimensiune mai mare, vom folosi numai primii 512 bytes.

Informațiile despre partițiile existente pe un disc sunt stocate într-o structură numită *tabelă de partiții*.

Tabela de partiții

De remarcat faptul că tabela de partiții a unui disc fizic este parte componentă a programului care se află pe primul sector al unui disc și are dimensiune și locație fixă, indiferent de tipul discului utilizat.

Tabela de partiții este localizată, în cadrul primului sector, la poziția hexazecimală **1BE** și are patru intrări. De fapt, tabela de partiții este un tablou unidimensional care constă din patru structuri identice, fiecare structură conținând informații despre o singură partiție.

Fiecare structură care descrie o partiție are dimensiunea de 16 octeți, deci cele patru structuri încep la pozițiile hexazecimale **1BE**, **1CE**, **1DE** și **1EE**.

Tabela de partiții este urmată de un cuvânt (2 octeți) care are valoarea hexazecimală **55AA**. Acest cuvânt este reprezentat de ultimii 2 octeți ai unui sector. Dacă, pe un disc, ultimul cuvânt nu are valoarea hexazecimală **55AA**, atunci înseamnă că nu a fost definită încă o tabelă de partiții sau discul a fost deteriorat.

Informații despre o partiție

Înainte de a începe prezentarea structurii care conține informații despre o partiție vă amintesc faptul că geometria unui disc este dată de patru parametri și anume:

- *număr de cilindri* – numărul de piste ale discului;
- *număr de capete de citire* – numărul capetelor de citire pentru fiecare cilindru;
- *număr de sectoare* – numărul de sectoare pentru fiecare cap de citire;
- *număr de octeți pentru fiecare sector* – dimensiunea în octeți a unui sector.

Capacitatea de stocare a unui disc este dată de produsul celor patru parametri prezentați anterior.

Structura care conține informații despre o anumită partiție este formată din mai multe câmpuri. Acestea sunt (în ordinea localizării lor):

- *indicatorul de boot* – acest câmp are dimensiunea de un octet, ne spune dacă de pe partiția respectivă există sistem de operare (are instalat un sistem de operare) și poate avea doar valorile 0 sau 128; dacă se întâlnește o valoare diferită de 0 sau 128 însemnă că partiția este invalidă; de obicei, cel mult o partiție de pe disc are acest câmp setat pe valoarea 128, și anume partiția de pe care se dorește încărcarea sistemului de operare;
- *numărul capului de început* pentru partiția curentă – are dimensiunea de un octet;
- *numărul sectorului de început* pentru partiția curentă - are dimensiunea de 6 biți, reprezintă numărul de ordine al sectorului cu care încep datele partiției în cadrul capului de început și poate avea valoarea egală cu cel puțin 1 și cel mult 64, el fiind reprezentat în cadrul structurii prin *valoare_reală* - 1; acest câmp este stocat pe cei mai din stânga șase biți ai celui de-al treilea octet al structurii curente;
- *numărul cilindrului de început* pentru partiția curentă - are dimensiunea de 10 biți, reprezintă numărul de ordine al cilindrului de la care începe partiția; acest câmp este format din cei mai din dreapta doi biți ai celui de-al treilea octet al structurii și din cel de-al patrulea octet al acestei structuri;
- *tipul partiției* - acest câmp ne indică tipul de date care sunt stocate în cadrul partiției și are dimensiunea de un octet;
- *numărul capului de sfârșit* pentru partiția curentă – are dimensiunea de un octet;
- *numărul sectorului de început* pentru partiția curentă - are dimensiunea de 6 biți, reprezintă numărul de ordine al sectorului cu care se termină datele partiției, în cadrul capului de început și poate avea valoarea egală cu cel puțin 1 și cel mult 64, el fiind reprezentat în cadrul structurii prin *valoare_reală* - 1; acest câmp este stocat pe cei mai din stânga șase biți ai celui de-al șaptelea octet al structurii curente;
- *numărul cilindrului de început* pentru partiția curentă - are dimensiunea de 10 biți, reprezintă numărul de ordine al cilindrului de la care începe partiția; acest câmp este format din cei mai din dreapta doi biți ai celui de-al șaptelea octet al structurii și din cel de-al optulea octet al acestei structuri;
- *numărul de sectoare care preced partiția* - acest câmp ne indică poziția partiției curente, reprezentată prin numărul de sectoare, relativ la începutul discului și are dimensiunea de 4 octeți;
- *numărul de sectoare folosite de partiție* - reprezintă numărul de sectoare alocate partiției și are dimensiunea de 4 octeți.

În cadrul unei partiții sunt stocate tot felul de date. Cel mai adeasea sunt stocate sisteme de fișiere. Probabil că ați auzi de denumiri ca *FAT*, *FAT32*, *NTFS*, *EXT2*. Ei bine, acestea sunt denumiri de sisteme de fișiere. Fiecare sistem de operare are propriul sistem de fișiere.

FAT, *FAT32* și *NTFS* sunt sisteme de fișiere care caracterizează sistemele de operare realizate de compania *Microsoft*, iar *EXT2* este un sistem de fișiere care este recunoscut de sistemele de operare din familia **NIX*.

Pe lângă sisteme de fișiere, partițiile conțin și alte tipuri de date, cum este cazul memoriei virtuale utilizată de sistemele de operare din familia **NIX*.

Un alt tip de dată stocat într-o partiție este tipul de partiție extinsă care conține informații despre alte partiții aflate pe disc.

Toate aceste tipuri de date au identificator unic (*tipul partiției*) pentru ca programul care încarcă un sistem de operare să sesizeze dacă tipul de dată care se află pe o anumită partiție este cel corect sau nu.

Din cauză că pe un disc se pot defini cel mult patru partiții, toate câmpurile intrărilor care nu conțin o partiție trebuie inițializate cu valoarea 0.

În continuare se află o listă de identificatori, scriși în baza 16, pentru diferite tipuri de partiții:

- 0x01 - *FAT*12;
- 0x04 - sistem de fișiere *FAT*16 cu dimensiunea mai mică de 32 *MB*;
- 0x05 - partiție extinsă;
- 0x06 - sistem de fișiere *FAT*16 cu dimensiunea mai mare de 32 *MB*;
- 0x0B - *FAT*32;
- 0x0C - *FAT*32 cu specificarea că accesarea se va face folosind numărul absolut al sectorului;
- 0x0E - *FAT*16 cu specificarea că accesarea se va face folosind numărul absolut al sectorului;
- 0x0F - partiție extinsă cu specificarea că accesarea se va face folosind numărul absolut al sectorului;
- 0x82 - memorie virtuală pentru sisteme de operare din familia **NIX*;
- 0x83 - *EXT*2.

Partiții extinse

Datorită faptului că tabela de partiții nu ne permite să avem mai mult de patru partiții s-a realizat un tip particular de partiție, și anume, *partiția extinsă*.

O partiție extinsă reflectă oarecum structura unui disc. În cadrul spațiului alocat pentru o partiție extinsă se mai pot defini alte partiții care poartă denumirea de partiții virtuale.

Primul sector al unei partiții extinse conține o tabelă de partiții similară celei de pe primul sector al discului. Diferențele dintre această tabelă de partiții și cea de pe primul sector sunt acelea că primul câmp al fiecărei intrări are întotdeauna valoarea 0 și câmpurile care reprezintă *numărul de sectoare care preced partiția* și *numărul de sectoare folosite de partiție* sunt relative la primul sector al partiției extinse.

În figura 2 se poate observa că utilizarea partițiilor extinse conduce la apariția unui arbore de partiții.

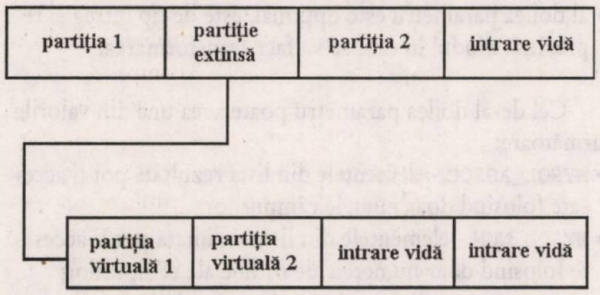


Figura 2: Exemplu de partiții virtuale

Partițiile definite în cadrul unei partiții extinse poartă denumirea de *partiții virtuale* tocmai din motivul că, dacă partiția extinsă este ștearsă, atunci partițiile definite în cadrul acesteia nu mai sunt accesibile.

În cadrul unei partiții extinse se poate defini o altă partiție extinsă.

O partiție care este definită în cadrul tabelei de partiții care se află pe primul sector al unui disc poartă denumirea de *partiție primară*. De obicei, denumirea de *partiție extinsă* se folosește pentru *partițiile virtuale*.

Limite ale dimensiunilor partițiilor

Datorită sistemelor de calcul mai vechi au existat câteva limite importante ale dimensiunilor partițiilor și, implicit, a dimensiunilor discurilor recunoscute de acestea.

Există mai multe moduri în care pot fi accesate informațiile stocate pe un disc prin intermediul sistemului standard de intrare / ieșire (*BIOS*).

Primul mod de acces este utilizarea parametrilor care reprezintă *numărul cilindrului*, *numărul capului* și *numărul sectorului care* se dorește a fi accesat.

Acest tip de acces a unui disc a dus la apariția unor limite de capacitate a partițiilor. Cea mai importantă limită a fost cea de 8 *GB* care a fost dată de produsul celor mai mari valori care pot fi stocate în cadrul tabelei de partiții pentru parametri *număr de cilindri*, *număr de capete de citire* și *număr de sectoare* având un disc ale cărui sectoare au dimensiunea de 512 octeți. Au mai fost și alte limite date de erori de programare. Acestea sunt limita de 512 *MB* și limita de 2 *GB*.

Al doilea mod de acces este cel al utilizării *numărului absolut al sectorului* care se dorește accestat. Datorită faptului că numărul de sectoare de pe disc se reprezintă folosind un număr întreg fără semn pe 32 de biți, dimensiunea unei partiții este limitată la 2048 *GB* = 2 *TB*.

Se pare că, deocamdată, nu există nici un fel de probleme cu privire la această limită deoarece, încă nu au apărut pe piață discuri având o astfel de capacitate.

O altă problemă care a dus la limitări asupra dimensiunilor partițiilor a fost dată de sistemele de fișiere. De exemplu, sistemul de fișiere *FAT* are o capacitate maximă de stocare de 2*GB*.

Datorită procesului rapid de dezvoltare tehnică, în viitor vom asista la o schimbare a acestor standarde legate de tabelele de partiții.

Pentru a partiționa discuri de diferite dimensiuni și tipuri există aplicații performante care oferă o multitudine de facilități, cum ar fi redimensionarea unei partiții, schimbarea tipului unei partiții, concatenarea sistemelor de fișiere de pe două partiții adiacente etc.

Claudiu Soroiu este student în anul IV la Universitatea Babeș-Bolyai din Cluj-Napoca și redactor-șef al GInfo. El poate fi contactat prin e-mail la adresa csoroiu@yahoo.com.

Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

În cadrul acestui articol vom continua prezentarea funcțiilor utilizate pentru accesarea bazelor de date MySQL și vom prezenta, de asemenea, modul de utilizare a poștei electronice folosind funcții oferite de limbajul PHP.

Baze de date MySQL

În numărul anterior am prezentat funcțiile de acces la bazele de date *MySQL* până la funcția `mysql_info`. În cele ce urmează vom continua prezentarea altor funcții utile în lucrul cu baze de date *MySQL*.

Funcția mysql_affected_rows

În cazul interogărilor în urma cărora nu se obține nici un rezultat, se poate folosi funcția `mysql_affected_rows` pentru a verifica numărul de înregistrări care au fost actualizate la ultima interogare a bazei de date, pentru celelalte tipuri de interogări putându-se folosi funcția `mysql_num_rows`.

Funcția `mysql_affected_rows` are un singur parametru care reprezintă identificatorul de acces al unei conexiuni spre un *server MySQL* și returnează un număr întreg, care reprezintă numărul de înregistrări actualizate la ultima interogare a bazei de date.

În continuare prezentăm un exemplu de utilizare a acestei funcții, iar în figura 1 poate fi observat efectul execuției *script*-ului:

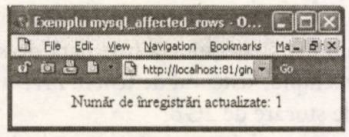


Figura 1: Funcția mysql_affected_rows

```
<HTML>
  <HEAD>
    <TITLE>
      Exemplu mysql_affected_rows
    </TITLE>
  </HEAD>
  <BODY>
  <?PHP
    $conn = mysql_connect("blue", "ginfo",
                                    "pginfo");
    if ($conn === FALSE)
      die("Conectare eșuată.");
    mysql_select_db("ginfo", $conn);
    mysql_query("update utilizatori set nume
      = 'soroiu' where nume = 'Soroiu'", $c...);
    $s = mysql_affected_rows($conn);
    echo "Număr de înregistrări actualizate: $s";
    mysql_close($conn);
  ?>
  </BODY>
</HTML>
```

Funcția mysql_fetch_array

Această funcție transformă o înregistrare dintr-un rezultat primit de la *server*-ul *MySQL* într-o listă.

Funcția `mysql_fetch_array` colectează datele referitoare la înregistrarea curentă și le returnează sub forma unei liste ale cărui elemente pot fi accesate folosind numele câmpurilor, numărul lor de ordine sau folosind ambelele moduri.

Dacă nu mai sunt înregistrări de prelucrat, funcția returnează valoarea logică **FALSE**.

Această funcție are doi parametri:

- primul parametru este de tipul resursă, a cărui valoare a fost setată folosind una dintre funcțiile `mysql_query` sau `mysql_unbuffered_query`;
- al doilea parametru este opțional, este de tip întreg și reprezintă modul în care se va face transformarea.

Cel de-al doilea parametru poate avea una din valorile următoare:

- `MYSQL_ASSOC` - elementele din lista rezultată pot fi accesate folosind doar numele câmpurilor;
- `MYSQL_NUM` - elementele din lista rezultată pot fi accesate folosind doar numerele de ordine ale câmpurilor;
- `MYSQL_BOTH` - elementele din lista rezultată pot fi accesate folosind ambele moduri prezentate anterior;

Funcția mysql_fetch_assoc

Această funcție transformă o înregistrare dintr-un rezultat primit de la *server*-ul *MySQL* într-o listă și are același efect cu cel al folosirii funcției `mysql_fetch_array`, folosind pentru cel de-al doilea parametru valoarea `MYSQL_ASSOC`.

Funcția mysql_fetch_num

Această funcție transformă o înregistrare dintr-un rezultat primit de la *server*-ul *MySQL* într-o listă și are același efect cu cel al folosirii funcției `mysql_fetch_array`, folosind pentru cel de-al doilea parametru valoarea `MYSQL_NUM`.

Funcția mysql_stat

Această funcție returnează un șir de caractere care reprezintă statusul *server*-ului *MySQL* și are un singur parametru care reprezintă identificatorul de acces la o conexiune către un *server MySQL*.

În continuare prezentăm un exemplu de utilizare a acestei funcții, iar în figura 2 poate fi observat efectul execuției acestui *script*:

```
<HTML>
  <HEAD>
    <TITLE>
      Exemplu mysql_stat
    </TITLE>
  </HEAD>
  <BODY>
  <PRE>
  <?PHP
    $conn = mysql_connect("blue","ginfo",
                                    "pginfo");
    $status = explode('  ',mysql_stat($conn));
    print_r($status);
    mysql_close($conn);
  ?>
  </PRE>
  </BODY>
</HTML>
```

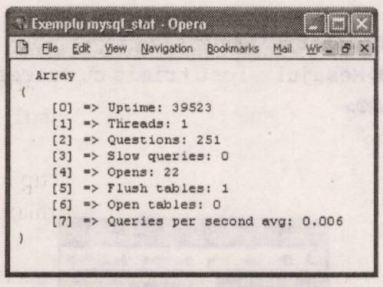


Figura 2: Funcția mysql_stat

Setul de funcții de acces la bazele de date *MySQL* oferit de limbajul *PHP* este vast și în cadrul acestei serii de articole ne-am rezumat la prezentarea celor mai importante astfel de funcții. Cu ajutorul acestor funcții se poate administra și utiliza foarte ușor orice bază de date *MySQL* indiferent de complexitatea și dimensiunea lor.

Poșta electronică

Faptul care a dus la utilizarea pe scară largă a acestui limbaj de *script*-ing se datorează multitudinii de funcții pe care acesta le oferă.

Biblioteca oferită de limbajul *PHP* conține și funcții utile pentru transmiterea și prelucrarea mesajelor de poștă electronică.

Limbajul *PHP* are integrat un client pentru protocolul *SMTP* (*Simple Mail Transfer Protocol*) și un set de funcții necesare pentru prelucrarea mesajelor primite de către un *server* de *mail* în vederea utilizării datelor conținute.

Cu ajutorul acestor funcții se pot realiza aplicații interesante cum sunt administrarea listelor de discuții, transmiterea unor știri tuturor utilizatorilor înscriși pe o anumită listă, difuzarea de noutăți cu privire la diverse apariții etc.

În continuare vă prezentăm funcția `mail`, cea mai importantă funcție pe care limbajul *PHP* o oferă pentru transmiterea mesajelor de poștă electronică.

Funcția mail

Această funcție se folosește pentru a transmite un mesaj de poștă electronică la una sau mai multe căsuțe poștale.

Funcția `mail` are cinci parametri de tip șir de caractere, dintre care ultimii doi sunt opționali.

Primul parametru reprezintă o listă a tuturor căsuțelor poștale, separate prin caracterul ",", care reprezintă destinatarii mesajului care se va transmite. O căsuță poștală din lista destinatorilor poate fi specificată utilizând unul dintre următoarele formate "`nume_destinatar <adresă_email>`" (fără ghilimele) sau "`adresă_email`" (fără ghilimele), unde `nume_destinatar` reprezintă numele proprietarului căsuței poștale `adresă_email` și poate fi și șirul vid, iar `adresă_email` reprezintă adresa căsuței poștale și are formatul "`nume_utilizator@domeniu`" (fără ghilimele), unde `domeniu` reprezintă numele *server*-ului care găzduiește căsuța poștală a utilizatorului `nume_utilizator`.

În cazul utilizării primului format, pentru a specifica un destinatar pentru mesajul care se va transmite, este obligatorie folosirea caracterelor "<" și ">" înainte, respectiv după câmpul `adresă_email`, iar dacă valoarea câmpului `nume_destinatar` conține caracterul spațiu este necesară utilizarea caracterului "" înaintea acestuia și după acesta. De exemplu, adresa "`"Claudiu Soroiu" <csoroiu@yahoo.com>`" este o destinație validă pentru un mesaj de poștă electronică.

Al doilea parametru reprezintă subiectul mesajului care se va transmite.

Cel de-al treilea parametru reprezintă conținutul mesajului care va fi transmis.

Pentru a putea trimite mesaje care conțin atașamente sau care au un tip de conținut altul decât text (ex. *HTML*)

este necesar ca valoarea acestui parametru să fie în format *MIME* (*Multi-purpose Internet Mail Extension*).

Cel de-al patrulea parametru reprezintă o listă de linii de antet suplimentare, separate prin șirul de caractere "\r\n" (sfârșit de linie), pentru mesajul care va fi transmis.

Ultimul parametru reprezintă o listă de opțiuni suplimentare pentru aplicația care va fi folosită pentru a efectua trimiterea mesajului și este dependent de aceasta.

Funcția returnează valoarea logică **TRUE** dacă s-a reușit trimiterea mesajului și valoarea logică **FALSE** în caz contrar.

În continuare vă prezentăm un exemplu de utilizare a acestei funcții.

```
<?PHP
  if (isset($_POST["trimite"])){
    $a = mail($_POST["email"],
          $_POST["subject"], $_POST["mesaj"]);
  if ($a)
    include "mesajtrimis.html";
  else
    include "mesajnetrimis.html";
  }
  else
    include "formular.html";
?>
```

Script-ul anterior realizează transmiterea unui *e-mail* pe baza datelor dintr-un formular *HTML*. În momentul încărcării *script*-ului, acesta afișează pagina "formular.html" care reprezintă un formular similar cu cel din figura 3.

Codul corespunzător formularului din fișierul "formular.html" este prezentat în continuare:

```
<HTML>
  <HEAD>
    <TITLE>
      Trimitere mesaj
    </TITLE>
  </HEAD>
  <BODY>
    <FORM method = "post" action =
                              "mailsender.php">
      Subiect:<BR>
      <INPUT type = "text" name = "subject">
      <P>
      Mesaj:<BR>
      <TEXTAREA cols = "40" rows = "10" name =
                                    "mesaj">
      </TEXTAREA>
      <P>
      Adresa de mail:<BR>
    <INPUT type = "text" name = "email">
    <P>
    <INPUT type = "submit" name = "trimite"
                          value = "Trimite">
    </FORM>
  </BODY>
</HTML>
```

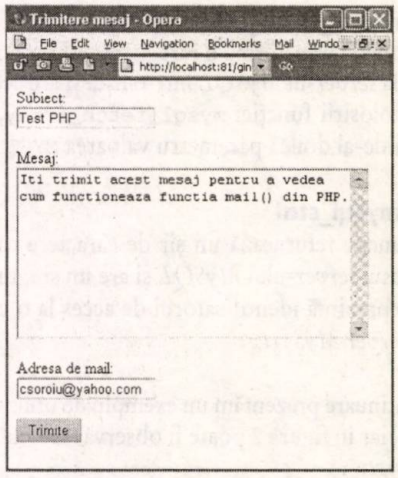


Figura 3: Formular de trimitere a unui mesaj

În momentul apăsării butonului "Trimite", *script*-ul încearcă trimiterea mesajului cu subiectul "Test PHP" și conținutul specifica în cadrul câmpului mesaj, la adresa csoroiu@yahoo.com.

Se observă că am folosit al doilea format prezentat pentru a specifica destinatarul mesajului.

Dacă se reușește trimiterea mesajului, atunci *script*-ul afișează pagina "mesajtrimis.html" după cum se poate observa în figura 4, iar în continuare se află codul *HTML* pentru această pagină:

```
<HTML>
  <HEAD>
    <TITLE>
      Mesaj trimis
    </TITLE>
  </HEAD>
  <BODY>
    <FONT color = "navy" size = "+1">
      <B>Mesajul a fost trimis cu succes.</B>
    </FONT>
  </BODY>
</HTML>
```

Figura 4: Confirmare trimitere mesaj

În cazul în care nu se reușește trimiterea mesajului, atunci *script*-ul afișează pagina "mesajnetrimis.html"

după cum se poate observa în figura 5, iar în continuare se află codul *HTML* pentru această pagină:

```
<HTML>
  <HEAD>
    <TITLE>
      Eroare trimitere mesaj
    </TITLE>
  </HEAD>
  <BODY>
    <FONT color = "red"  size = "+1">
      <B>Mesajul nu a putut fi trimis.</B>
    </FONT>
  </BODY>
</HTML>
```

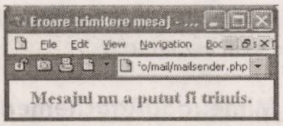


Figura 5: Eroare trimitere mesaj

PEAR

PEAR (*PHP Extension and Application Repository*) reprezintă un proiect demarat de echipa de programatori *PHP*. Scopul acestui proiect este de a oferi o bibliotecă structurată de clase *open-source* tuturor utilizatorilor limbajului *PHP*.

Această bibliotecă oferă, printre multe altele, un set de clase care pot fi folosite pentru a putea transmite mesaje electronice care să conțină atașamente și al căror conținut poate fi în format *HTML* în loc de text.

Acest set de clase poate fi descărcat de la adresa *web* **http://pear.php.net/Mail_Mime**. Pentru a putea folosi setul de clase *Mail_Mime* trebuie instalat mai întâi pachetul *PEAR* de bază care poate fi găsit la adresa **http://pear.php.net**.

În continuare vom prezenta clasa `Mail_Mime`, cea mai importantă clasă necesară pentru a codifica în format *MIME* astfel de mesaje.

Clasa Mail_mime

Această clasă se folosește pentru a construi mesaje electronice în format *MIME*.

Puțină configurare PHP...

Pentru a putea utiliza funcțiile oferite de limbajul PHP pentru a trimite mesaje de poștă electronică, pe sistemele de operare din familia Windows sau *NIX este necesar să se seteze corect interpretorul PHP prin intermediul fișierului php.ini localizat în directorul în care este instalat sistemul de operare, respectiv în directorul /etc pe sisteme *NIX.

Funcția mail trimite mesajele prin intermediul unui server SMTP (Simple Mail Transfer Protocol) care poate fi găzduit de server-ul pe care este instalat interpretorul PHP sau de către un alt server accesibil interpretorului.

În fișierul php.ini trebuie adăugate variabilele având numele SMTP, smtp_port, sendmail_from și sendmail_path, dacă acestea nu există, sau trebuie modificate valorile acestora.

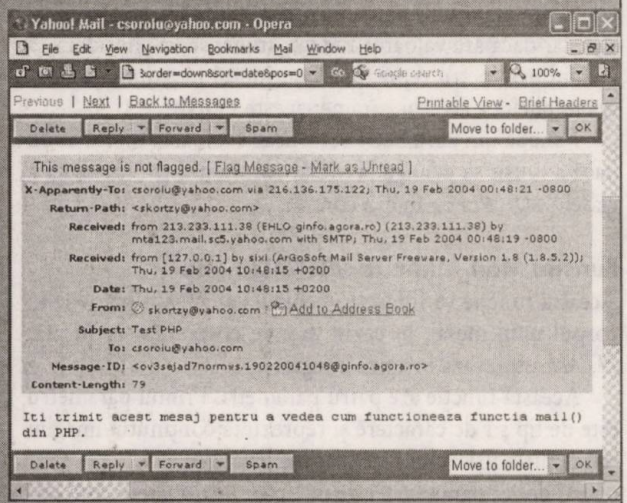
Variabila SMTP reprezintă ip-ul sau adresa server-ului de mesaje electronice.

Dacă această variabilă lipsește, atunci, pentru sistemele de operare din familia Windows se va încerca folosirea sistemului pe care se află instalat interpretorul pentru a transmite mesaje electronice, iar pentru sistemele de operare din familia *NIX se vor folosi setările implicite ale sistemului.

Variabila smtp_port reprezintă portul pe care răspunde server-ul de mesaje electronice și are valoarea implicită 25, pentru sistemele de operare din familia Windows sau setarea implicită a sistemului, pentru sistemele de operare din familia *NIX.

Variabila sendmail_from reprezintă adresa de mail a expeditorului. După cum se observă în imaginea următoare am folosit adresa skortzy@yahoo.com. Valoarea acestei variabile este folosită numai pentru sistemele de operare din familia Windows, iar pe sistemele de operare din familia *NIX se folosește setarea implicită a sistemului.

Variabila sendmail_path reprezintă calea către aplicația care se ocupă de transmiterea mesajelor pe sistemele de operare din familia *NIX.



Mesajul primit pe adresa csoroiu@yahoo.com

Constructorul clasei

Clasa `Mail_Mime` are șase funcții membru a căror funcționalitate o vom prezenta în continuare și un constructor

Constructorul `Mail_Mime` creează o instanță a clasei și are un singur parametru de tip șir de caractere, care este opțional. Acest parametru reprezintă terminatorul de linie care va fi utilizat de către clasă pentru a încheia fiecare linie din mesaj și are valoarea implicită `"\r\n"`.

Funcția Mail_Mime::setTxtBody

Această funcție setează conținutul text al unui mesaj pentru ca acesta să poată fi vizualizat în cadrul unui client de *mail* de tip text cum este cazul aplicației `mail` prezentă pe sistemele de operare din familia **NIX*.

Această funcție are doi parametri.

Primul parametru este de tip șir de caractere și reprezintă conținutul mesajului sau numele fișierului care conține mesajul în format text.

Cel de-al doilea parametru este de tip logic și este opțional. Dacă acest parametru are valoarea **TRUE**, atunci se va considera că primul parametru reprezintă numele unui fișier, iar dacă are valoarea **FALSE**, atunci se va considera că primul parametru reprezintă conținutul mesajului. Valoarea implicită a acestui parametru este **FALSE**.

Funcția `setTxtBody` returnează valoarea logică **TRUE** dacă a reușit să adauge la mesaj conținutul și o instanță a clasei `PEAR_Error` în caz contrar.

Funcția Mail_Mime::setHTMLBody

Această funcție setează conținutul *HTML* al unui mesaj pentru ca acesta să poată fi vizualizat în cadrul unui client de *mail* de tip *HTML* cum este cazul aplicației `Outlook` prezentă pe sistemele de operare din familia *Windows*.

Această funcție are doi parametri.

Primul parametru este de tip șir de caractere și reprezintă conținutul mesajului sau numele fișierului care conține mesajul în format *HTML*.

Cel de-al doilea parametru este de tip logic și este opțional. Dacă acest parametru are valoarea **TRUE**, atunci se va considera că primul parametru reprezintă numele unui fișier, iar dacă are valoarea **FALSE**, atunci se va considera că primul parametru reprezintă conținutul mesajului. Valoarea implicită a acestui parametru este **FALSE**.

Funcția `setHTMLBody` returnează valoarea logică **TRUE** dacă a reușit să adauge la mesaj conținutul și o instanță a clasei `PEAR_Error` în caz contrar.

Funcția Mail_Mime::addHTMLImage

Această funcție se folosește pentru a adăuga o imagine în corpul unui mesaj, în cazul în care conținutul în format *HTML* utilizează imagini integrate.

Această funcție are patru parametri. Primul parametru este de tip șir de caractere și reprezintă conținutul imagini sau numele fișierului care conține imaginea.

Al doilea parametru este de tip șir de caractere, este opțional și reprezintă tipul conținutului. Acest parametru are valoarea implicită `"application/octet-stream"`. De exemplu, dacă imaginea este de tip *GIF*, acest parametru poate primi valoarea `"image/gif"`.

Cel de-al treilea parametru este de tip șir de caractere, este opțional și reprezintă numele care va fi dat imaginii. În cazul în care primul parametru conține imaginea, atunci este necesar ca acest parametru să apară, în caz contrar se va folosi numele fișierului transmis ca valoare pentru primul parametru.

Cel de-al patrulea parametru este de tip logic și este opțional. Dacă acest parametru are valoarea **TRUE**, atunci se va considera că primul parametru reprezintă numele unui fișier, iar dacă are valoarea **FALSE**, atunci se va considera că primul parametru reprezintă conținutul unei imagini. Valoarea implicită a acestui parametru este **TRUE**.

Funcția `addHTMLImage` returnează valoarea logică **TRUE** dacă a reușit să adauge la mesaj conținutul și o instanță a clasei `PEAR_Error` în caz contrar.

Funcția Mail_Mime::addAttachement

Această funcție se folosește pentru a adăuga un fișier la un mesaj și are cinci parametri, dintre care primii patru au aceeași semnificație cu cei de la funcția precedentă.

Ultimul parametru este de tip șir de caractere, este opțional și reprezintă tipul codificării care va fi utilizată. Acest parametru are valoarea implicită `"base64"`. Pentru fișiere atașate care nu sunt binare (ex. fișiere executabile, arhive etc.) se poate folosi valoarea `"quoted-printable"`.

Funcția Mail_Mime::headers

Această funcție se folosește pentru a adăuga linii de antet suplimentare la mesajul constituit de această clasă.

Funcția `headers` are un singur parametru de tip listă, care are ca și chei numele câmpurilor antetelor și ca valori, valorile acestor câmpuri.

Funcția returnează un o referință la o listă de același tip care conține, pe lângă elementele transmise ca parametru, și elementele adăugate de această clasă, pentru ca mesajul să fie recunoscut cum trebuie de către un client de *mail*.

Această funcție nu poate fi apelată după apelul funcției `Mail_Mime::get`.

Funcția Mail_Mime::get

Această funcție se folosește pentru a obține conținutul mesajului electronic în format *MIME*, după ce acesta a fost construit.

Funcția `get` nu are nici un parametru și returnează un șir de caractere care reprezintă conținutul mesajului electronic codificat în format *MIME*.

Pentru a transmite un mesaj folosind funcția `mail` prezentată și această clasă, trebuie prelucrat șirul liniilor de antet returnat de funcția `Mail_Mime::headers` astfel încât să se obțină un șir de caractere, iar conținutul mesajului va fi constituit de șirul de caractere rezultat în urma apelării funcției `Mail_Mime::get`.

o publicație Agora Media

Publicarea acestei reviste este posibilă datorită companiilor de mai jos care sprijină învățământul românesc de informatică

SPONSORI DE PLATINĂ

FORTE COMPUTERS
Flexibility. Made in Romania

IBM®

sat
IT SOLUTIONS & SERVICES

hp®

net consulting

FUJITSU COMPUTERS SIEMENS

SPONSOR DE AUR

CCT
CABINETUL CĂLIN TATOMIR

Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-266.516

Martie 2004
Vol. 14/3

Lei 30.000

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

Babel

OPTIMIZAREA programelor

- sfaturi pentru viitorii olimpici

Plus

interviu
Despre ACM - reprezentanții României

concurs
Rundele 26-40 Top 50 după 25 runde

internet
PHP 4 - procesarea documentelor XML

5-8 iunie 2004

CUPA SIVECO

Concurs național
de software educațional

Sub patronajul Ministerului Educației, Cercetării și Tineretului
Partener media: PC Magazine România

trimiterea lucrărilor - până la 15 mai 2004

SIVECO ROMANIA SA

PC MAGAZINE ROMÂNIA

MINISTERUL EDUCAȚIEI,
CERCETĂRII ȘI TINERETULUI

Vol. 14, Nr. 3 /2004

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Director Bursele Agora:
Mihai Scorțaru
Redactor-șef:
Claudiu Soroiu
Redactor-șef adjunct:
Carmen Șușman
Redactori:
prof. *Eugen Ionescu*
Mihai Stroe
Colaboratori:
Cristian Olar
Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
400091, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/592.422, 0265/266.516
Fax: 0265/266.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359
Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

Director general:
Romulus Maier
Director executiv:
Adrian Pop

Vânzare reclamă:
Mihai Bucuroiu
Denis Marioara
Marius Comărniceanu
Difuzare:
Florin Dinu
Florin Oprescu
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134
Reprezentanța București:
Director: *Dana Pastramă*.
Tel: 021/330.92.81
Fax: 021/330.92.85

Sumar

actual.. PAG. 4

Pledoarie pentru JAVA
Mihai SCORȚARU

INFOMATRIX

Microsoft încurajează tinerii dezvoltatori
Alexandra FOAMETE

babel.. PAG. 36

OPTIMIZAREA programelor
Mihai STROE

~ metode de optimizare a ~
~ timpului de execuție ~

probleme.. PAG. 9

Concursul INTER-JUDEȚEAN de programare INFO OLTENIA

Faza JUDEȚEANĂ a olimpiadei de INFORMATICĂ

internet.. PAG. 40

Pagini WEB cu PHP 4
Mihai SCORȚARU
Claudiu SOROIU

~ procesarea ~
~ documentelor XML ~

concurs.. PAG. 23

Despre rundele #16-#25

Top 50 după 25 runde

Rundele #26-#40

interviu.. PAG. 47

Despre ACM
Mihai STROE

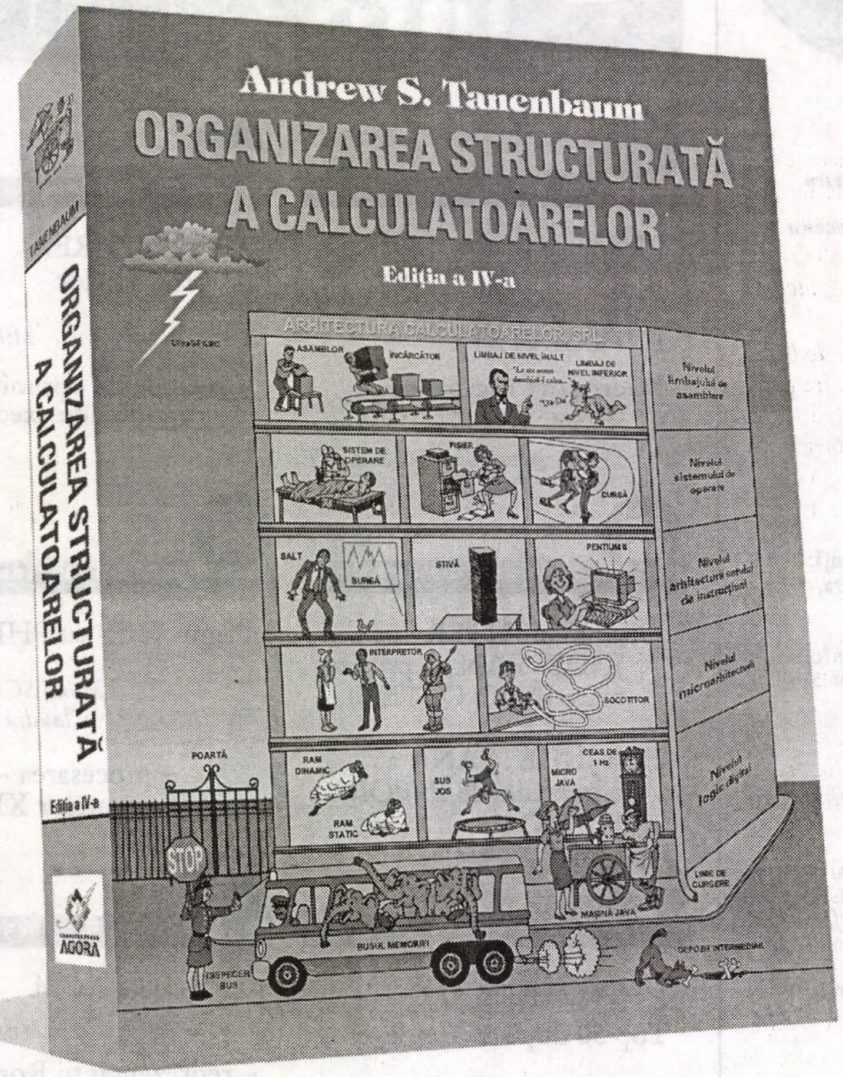
~ reprezentanții României ~
~ la finala ACM 2004 ~

```
class Olimpiada{
 public static void main(String args[]){
  System.out.println("Felicit[illegible]");
  for (int i=5; i < 12; i++){
   System.out.println("[illegible] + i + " - a")
  }
  System.out.[illegible] ( ONI 2004")
  System.out.println("[illegible]au 2-9 aprilie");
 }
}
```

Redacția GInfo urează mult succes tuturor participanților la ediția 2004 a Olimpiadei Naționale de Informatică!

Cine are carte, are parte!

...de înțelegerea calculatoarelor!



Comandă-o acum și dacă ești student primești și o **REDUCERE** semnificativă!

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65, 0721-927.499

A VENIT PRIMĂVARA...

Motto: Cu o floare nu se poate face primăvară,
Dar este un început promițător.

A venit primăvara și cu această ocazie au fost centralizate listele de participanți la *Olimpiada Națională de Informatică* ce au fost stabilite în urma desfășurării *Olimpiadei Județene de Informatică*.

Olimpiada Județeană de Informatică este o competiție la care participă elevi din toate județele țării care sunt capabili să rezolve probleme cu un grad mai ridicat de dificultate.

Se știe că gradul de dificultate al problemelor propuse spre rezolvare la concursurile de programare crește de la an la an.

De câțiva ani, la olimpiadele de informatică pot participa și elevi ai școlilor generale.

Subiectele propuse spre rezolvare la ediția din acest an a *Olimpiadei Județene de Informatică* au fost unice în toată țara.

Această ediție, ca de altfel și edițiile precedente, a adus, în egală măsură, atât bucurii, cât și dezamăgiri în rândurile participanților.

Cei care au reușit să se califice la *Olimpiada Națională de Informatică* și cei care, chiar dacă nu s-au calificat, dar au obținut rezultate mai bune decât s-au așteptat au fost foarte entuziasmați în urma acestui eveniment.

Concurenții care au fost nemulțumiți în urma concursului au avut mai multe motive, dar de cele mai multe ori au dat vina pe modul de corectare al soluțiilor lor, deci, în concluzie, pe profesorii care au făcut parte din comisiile de corectare.

Consider că nu a fost vina nici unui profesor care luat parte la desfășurarea concursului, deoarece, cel puțin în cazul problemelor în enunțul cărora se specifică faptul că rezultatele trebuie scrise într-un anumit fișier, evaluarea este un proces automat care constă în compilarea surselor concurenților, executarea programelor pe un set de teste, monitorizarea timpilor de execuție și validarea rezultatelor. *Foarte simplu!*

Se poate întâmpla ca, din greșeală, pentru un anumit concurent, să nu se evalueze soluțiile pentru probleme, dar, tocmai din acest motiv a fost introdusă o perioadă în care se pot face contestații. *Nimeni nu este perfect!*

În continuare voi încerca să explic ceea ce cred eu că se întâmplă în timpul concursului, și, în concluzie, ceea ce duce la obținerea unor punctaje mici, având în vedere că, personal am participat la diferite concursuri, mă ocup și fac parte din comisia de evaluare a soluțiilor problemelor propuse spre rezolvare la concursul de programare *Bursele Agora*.

Motivul care duce la apariția a ceea ce voi relata îl constituie criza de timp în care intră anumiți concurenți și emoțiile, deoarece, indiferent despre ce eveniment este vorba, toți avem emoții.

Datorită faptului că este incomod ca, în timpul rezolvării unei probleme, să se vizualizeze conținutul fișierului care conține rezultatul execuției programului, concurenții scriu rezultatul pe ecran și, uită efectiv să modifice modul de furnizare a rezultatului sau, dacă au făcut aceste modificări, din grabă scriu greșit numele fișierului.

O soluție în acest sens ar fi ca, rezultatele să fie scrise atât pe ecran, cât și în fișierul de ieșire precizat în enunțul problemelor.

De foate multe ori, nu sunt înțelese bine cerințele anumitor probleme și pentru aceasta recomand participanților neexperimentați să citească de mai multe ori enunțul.

Se întâmplă destul de des, ca anumite porțiuni din rezolvare, care sunt foarte simple, să fie scrise greșit din neatenție sau din simplul motiv, că pentru testarea unei rezolvări se folosesc doar exemplele din enunțul problemelor. Din această cauză nu se obține punctaj maxim la toate testele.

Recomand concurenților folosirea unor exemple cât mai variate pentru a fi siguri că programele furnizează rezultate bune.

Cea mai frecventă greșeală pe care o fac majoritatea concurenților este aceea că fac ultimele modificări în criză de timp și nu mai testează nici dacă programele se compilează și nici dacă furnizează soluțiile corecte. *Deci...*

Concursurile, în general, sunt evenimente la care nu trebuie neapărat să câștigăm, important este să participăm și să încercăm să ne depășim pe noi înșine. Nu contează că *Xulescu* este mai bun sau *Xulescu* este mai slab.

În final le urez *Baftă!* tuturor participanților la concursurile de informatică și *Atenție sporită!*.

Claudiu Soroiu

Pledoarie pentru JAVA

Mihai Scorțaru

Și de această dată am primit scrisori de la cititori care doresc să își exprime părerea despre introducerea predării limbajului Java în licee. Așteptăm în continuare replici la mesajele publicate în paginile revistei noastre. Mulțumim tuturor celor care ne-au scris...

Andrei Matei

elev, clasa a XII-a

Am citit cu interes articolul domnului Mihai Scorțaru despre limbajul Java și predarea sa în licee. Personal, nu știu dacă este o idee foarte bună înlocuirea C-ului cu Java. Întrucât în liceu se vrea predarea algoritmilor și a tehnicilor de programare, părerea mea este că introducerea limbajului Java ar complica "inutil" lucrurile, în sensul că ar deturna accentul de pe algoritm pe detalii de limbaj/mediu/ bibliotecă de clase/mașină virtuală. Cât despre partea cu actualitatea, cine știe cum va fi peste 4-5 ani. Deja a mai pierdut o importantă cotă de piață după abandonarea de către Microsoft și dezvoltarea în paralel a limbajului C#.

Oricum, nu asta era teza pe care vroiam să o susțin. În fond mi se pare un detaliu care este limbajul folosit. Părerea mea este că modul de a preda informatică în liceu este greșit conceptual. Sunt elev la o clasă cu profil de informatică intensiv la Colegiul Național Tudor Vianu din București.

În urma celor patru ani de studiu am rămas cu impresia că cele șase-șapte ore de informatică din fiecare săptămână nu au folosit la mare lucru. În primul rând nu cred că programarea și algoritmica ar trebui predate unui număr atât de mare de elevi cum sunt la mine în liceu și cred că în toată România. La urma urmei programarea este doar una din multele meserii posibile pentru un om (să zicem cu studii superioare).

Contabilitatea nu este predată în liceu, și probabil sunt mai mulți contabili decât programatori. Nici arhitectura nu este predată pe scară largă, nici construcția de poduri și nici depanarea de telefoane mobile.

Câți dintre zecile de mii de elevi care termină în fiecare an liceul după ce au făcut nenumărate ore de informatică vor trebui să programeze vreodată în viață? Nu cred că prea mulți.

În opinia mea programarea este predată la o scară mult prea largă. Mi-ați putea răspunde că o astfel de materie nu e predată exclusiv pentru a-i învăța pe elevi să programeze, ci și pentru a le dezvolta imaginația și un anumit fel de gândire. Sunt de acord, dar credeam că din același motiv se face și o tonă de matematică (că doar nu o să trebuiască să integrez de foarte multe ori în viață).

S-ar putea să îmi spuneți că nu e obligat nimeni să vină la clase de informatică intensiv. Da și nu... Aceste clase sunt considerate cele mai bune din licee și elevii simt presiunea societății să se ducă la o astfel de clasă. Și chiar și în programa claselor care nu poartă colosalul atribut de "intensiv" sunt prevăzute o mulțime de ore de informatică pe săptămână.

Cred că timpul folosit pentru a învăța "masele" programare ar putea fi folosit mult mai bine pentru a-i învăța pe elevi să folosească cât mai bine unelte și tehnologii de care se vor izbi inevitabil în câmpul muncii. Ar trebui predat Windows, Office, cât mai multe despre Internet, e-mail etc.

Pe de altă parte nici mica fracțiune de elevi care vor ajunge programatori nu prea profită de multele ore de informatică. Deși este materia cu cele mai multe ore pe săptămână, materia predată efectiv este extrem de sumară și diluată. Pot să spun că la școală nu se învață mare lucru, cel puțin în comparație cu ce trebuie să știe un programator.

Despre ținutul la zi cu ultimele tehnologii nici nu poate fi vorba, dar nici cine știe ce algoritmică nu se predă. Cred că cel mai avansat subiect abordat este algoritmul lui Dijkstra.

Nu se poate da în totalitate vina pe profesori. Unii dintre ei sunt cu adevărat dăruiți meseriei lor și talentați, dar nu se poate avansa prea mult cu grupe foarte mari de elevi dezinteresați (pe lângă excepțiile interesate).

Ca o concluzie aș zice că majoritatea colegilor mei au trecut prin materia informatică ca gâsca prin apă, au irosit șase ore în fiecare săptămână (aproape o zi de lucru!). Sunt sigur că mulți dintre ei nu sunt în stare acum (în clasa a XII-a) să inverseze elementele unei matrice sau să sorteze un vector. Poate ar trebui regândit modul în care se predă informatică în liceu și țelurile urmărite.

Răspunsul...

Pentru început trebuie să menționez faptul că eu nu consider potrivită ideea conform căreia în liceu ar trebui să se predea algoritmi și tehnici de programare.

Limbajul *Java* nu este potrivit pentru așa ceva; avantajele sale sunt în alte domenii. Totuși nu cred că ar complica lucrurile deoarece se poate prezenta sumar ce trebuie să conțină un program *Java* pentru a putea fi compilat și apoi se trece la prezentarea mediului sau cum se face acum pentru mediile *Pascal* și *C/C++*.

Nu cred că poate cineva să susțină că mediile *Java* sunt mai dificil de utilizat decât cele actuale. Părerea mea este că, dimpotrivă, aceste medii sunt incomparabil mai puternice.

Așa cum spuneam și într-o replică anterioară, folosirea limbajului *Java* nu ne obligă să renunțăm la algoritmi. Diferența este că cei care nu sunt interesați de algoritmi au mult mai multe alte opțiuni în *Java*.

Cred că numărul elevilor plictisiți care nu înțeleg și nu vor să înțeleagă ce este acela un arbore binar se va reduce deoarece cei neinteresați de algoritmi pot să experimenteze alte aspecte ale limbajului.

Despre partea cu actualitatea, nu a susținut nimeni să se introducă acum *Java* și să rămână așa timp de douăzeci de ani. Dacă limbajul va fi depășit, atunci va trebui înlocuit în programă.

Referitor la C#, eu văd lucrurile dintr-o altă perspectivă. Este adevarat că există diferențe între *Java* și C#, dar acestea sunt nesemnificative. De exemplu, eu cunoșteam doar limbajul *Java* și am fost nevoit să dezvolt o aplicație C#. Nu am avut absolut nici o problemă. Au existat doar diferențe de detaliu cu care m-am obișnuit după numai câteva ore de lucru.

Nu știu dacă există ceva numit "piață" a unui limbaj. Ceea ce știu sigur este că acum trei ani majoritatea programatorilor pe care îi cunosc programau în C++, iar acum majoritatea programează în *Java*. Este adevărat că a început să fie utilizat C# în detrimentul limbajului *Java* dar, datorită asemănării foarte mari dintre cele două limbaje, contează "cota de piață" a limbajelor *Java* și C# împreună care este mult mai mare decât era "cota de piață" a limbajului *Java* acum doi ani.

Cu celelalte afirmații nu pot decât să fiu de acord. Problema este că trebuie să se facă ceva. O altă problemă este că toată lumea așteaptă ca altcineva să facă ceva. Am câteva observații: contabilitate se mai predă prin licee (nu la toată lumea, dar se predă), gândirea algoritmică este, în opinia mea, foarte diferită de gândirea matematică și mulți programatori profită de orele de informatică din liceu chiar dacă nu își dau seama de acest fapt sau nu vor să o recunoască.

Majoritatea programatorilor susțin, pe bună dreptate că "au învățat singuri". Totuși, dacă nu ar fi fost acele ore de informatică în care li se părea că pierd vremea, sunt convins că mulți dintre ei nici nu ar fi încercat să învețe singuri.

Eu cred că orele de programare sunt necesare și nu cred că numărul elevilor de la clasele de informatică este prea mare. Totuși, ar trebui ca programa școlară să conțină ceva util pentru acești elevi. Și cum *Windows*, *Office* etc. se poate studia începând din clasa a V-a, în liceu rămâne programarea, iar orele de informatică trebuie să fie folositoare elevilor.

Cristian Bordeianu

profesor, Grupul Școlar nr.1, Cîmpulung Moldovenesc

Publicarea articolului dumneavoastră din decembrie 2003 a fost pentru mine o surpriză plăcută. Nu am trimis atunci opinia mea pentru că am așteptat părerile profesorilor cu mai multă experiență.

În ceea ce mă privește, deși formația mea de bază este de profesor de fizică, predau de trei ani și informatică. Asta după ce am urmat timp de doi ani cursuri post-universitare de specializare la Facultatea de Informatică a Universității Alexandru Ioan Cuza din Iași, deși mă aflu în al cincilea deceniu de viață.

Pot să spun însă că am fost dezamăgit de reacțiile profesorilor de informatică publicate în numărul din ianuarie 2004. Am așteptat o discuție sinceră privitoare la avantajele și dezavantajele limbajului Java și în loc de asta am găsit atitudini polemice privitoare la locul profesorului de informatică și în general a situației din învățământul liceal.

Sunt de acord cu majoritatea argumentelor prezentate mai ales în ceea ce privește programul AEL, dar aș prefera să continui discuția pe alte baze.

Astfel cred că principalele argumente pentru învățarea limbajului Java în liceu sunt:

- *Java este independent de platformă sau sistem de operare. Programul sursă poate fi folosit neschimbat pe orice calculator.*
- *Java este un limbaj de programare orientat pe obiecte.*
- *Bibliotecile standard ale limbajului Java oferă suport pentru grafică, iar codul grafic este și el portabil.*
- *Java facilitează utilizarea graficii interactive și a altor efecte speciale în paginile web. Anumite programe Java numite applets sunt executate în cadrul paginilor web.*
- *Rularea appleturilor Java pe Internet este sigură. Java*

este proiectat în așa fel încât appleturile executate în cadrul paginilor web să aibă acces limitat la computerul pe care sunt rulate.

- *Compilatorul și kitul de dezvoltare Java (SDK, JDK) pot fi descărcate gratuit de pe site-ul firmei Sun.*
- *Appleturile Java pot fi rulate cu ajutorul unui browser de pe Internet, ceea ce permite profesorilor să scrie programe demonstrative care să poată fi rulate de elevi sau studenți pe orice calculator conectat.*
- *Java este ușor de învățat, sintaxa fiind asemănătore cu cea a limbajului C/C++.*

Eu am învățat singur Java și cred că trecerea de la C/C++ la Java este destul de ușoară. În orice caz mult mai ușoară decât trecerea de la Turbo Pascal la C. Și opțiunea pentru Java ca prim limbaj de programare este demnă de luat în considerare, cel puțin luând în seamă rezultatele bune obținute în anumite universități de la noi.

Deși lucrez în Java de mai puțin de doi ani, am reușit să realizez o serie de appleturi cu aplicații în fizică care pot fi găsite pe site-ul: **http://www.geocities.com/cbord2000/**. *De asemenea, colaborez la scrierea unei cărți de fizică informatică care folosește tot limbajul Java:*

Landau R., Paez M., Bordeianu C., **Computational Physics with Java**, *John Wiley & Sons (va fi publicată în 2005).*

Menționez că cel puțin în domeniul fizicii informatice (Computational Physics) Java tinde să devină noul standard în SUA.

Închei cu speranța că toate acestea vor folosi la ceva și că vom reuși să învingem inerția în folosul elevilor.

Un (fel de) răspuns...

Întrucat opiniile dumneavoastră coincid în mare măsură cu ale mele, nu prea am ce să vă răspund. Așadar, prefer să îi las pe alții să o facă, dacă doresc. Eu vă mulțumesc mult pentru scrisoare și vă felicit pentru munca depusă. Mult succes în continuare!

O elevă

cls. a X-a

Eu sunt elevă în clasa a X-a la un liceu din țară.

Am citit toate articolele din GInfo din ultima vreme și îmi pare bine că s-a adus în discuție o astefel de temă.

Am de spus câteva lucruri în legătură cu aceasta. În primul rând, problema nu se referă la informatică și predarea informaticii în școală, ci se referă la învățământul românesc, și dacă vreți să o luăm mai departe și mai general se poate duce problema la situația în care se află țara noastră.

Este jalnic și trist că nu toți își dau seama. Unii trăiesc într-un vis continuu și nu au puterea să se ridice deasupra și să privească problema în ansamblu. Nu e vina lor (în totalitate). Se poate spune că tot SISTEMUL nu le dă voie.

Dar noi în loc să stăm aici și să dăm apă la moară cu aceste discuții, am putea folosi timpul mult mai eficient prin găsirea unei situații care să rezolve problema, fie ea numai în domeniul informaticii sau chiar numai în predarea informaticii în liceu sau la fel de bine în facultate.

Nu există probleme care să nu poată să fie rezolvate.

Eu propun ca discuția să se îndrepte spre o cale de a schimba ceea ce nu este bine și ceea ce nu merge cum trebuie.

Ar fi foarte interesant dacă s-ar putea închega un proiect de popularizare a unei idei (de exemplu Pro Java sau Pro POO).

Până la acest moment eu zic că este singura soluție care ar putea avea rezultate.

Multumesc mult și aștept cu nerăbdare replici...

Replica...

Într-adevăr, marea majoritate a românilor sunt de acord asupra faptului că "sistemul e de vină". Ceea ce nu se ia în vedere este faptul că sistemul este format din noi. Sistemul nu funcționează corect, deoarece marea majoritate a componentelor care îl formează nu funcționează corect.

Și cum nimeni nu vrea să riște, sistemul rămâne așa. Știm că ar trebui să fie mai bine și poate fi mult mai bine, dar poate nu merită efortul. Și pentru un român, dacă rezultatele nu sunt imediate, atunci nu merită efortul.

Eu nu cred că noi stăm aici și dăm apă la moară cu aceste discuții, irosind timp prețios care ar putea fi utilizat în alte scopuri. Noi, ca redacție, putem să aducem tema în discuție. Din nefericire, nu putem face mai mult deocamdată, deși am vrea.

Se tot spune în ultima vreme că tinerii trebuie să preia inițiativa. Din nefericire, există două probleme: în primul rând tinerii nu sunt lăsați să preia inițiativa; în al doilea rând (mult mai trist, după părerea mea) tinerii nu doresc să preia inițiativa.

Unii pot considera că nu există probleme care nu pot fi rezolvate. Nu cred că este adevărat! Este aproape copilăresc să credem că putem rezolva orice problemă, că putem mulțumi pe toată lumea. Ceea ce este important este să încercăm să mulțumim pe cât mai mulți.

Prin această discuție, dăm ocazia tuturor celor care doresc să schimbe ceva sau au făcut deja ceva să își exprime părerea, dacă vor. Nu mai putem aștepta să facă alții ceea ce trebuie făcut.

Închei cu un citat pe care l-am auzit destul de des în ultima vreme și care îmi place foarte mult:

"Libertatea de a lupta pentru binele comun este un drept de care puțini au curajul să se bucure!".

INFOMATRIX

În luna noiembrie 2003, Lumina Instituții de Învățământ, reprezentată de cele două școli ale sale: International School of Bucharest și Liceul Internațional de Informatică București, în colaborare cu Ministerul Educației, Cercetării și Tineretului și Inspectoratul Școlar al Municipiului București au anunțat începerea celei de-a doua ediții a InfoMatrix, un proiect informatic de mare succes, destinat elevilor de gimnaziu și liceu.

Anul 2002 reprezintă punctul de plecare al *InfoMatrix*. Ediția națională a concursului a fost apreciată de elevi. Peste 200 de proiecte au fost înregistrate, iar 40 s-au calificat în faza finală. Anul acesta concursul se dorește a fi o competiție internațională. Până acum s-au înscris participanți din 33 de țări cu 267 de proiecte. Țările participante sunt: *Africa de Sud*, *Albania*, *Arabia Saudită*, *Azerbaidjan*, *Bosnia și Herzegovina*, *Cipru*, *Danemarca*, *Georgia*, *India*, *Kazakhstan*, *Kenya*, *Kârgâstan*, *Republica Macedoneană*, *Insulele Mauritius*, *Mexic*, *Pakistan*, *Peru*, *Polonia*, *Republica Moldova*, *România*, *Federația Rusă*, *Slovacia*, *Spania*, *Suedia*, *Republica Tataristan*, *Thailanda*, *Turcia*, *Turkmenistan*, *Ucraina*, *Uzbekistan*, *Venezuela*, *Yemen*.

LUMINA
Institutii de Invatamant
Educational Institutions

După cum am spus, *InfoMatrix* este un concurs internațional de informatică, destinat tinerei generații - elevilor de gimnaziu și liceu. Am ales această categorie de vârstă 13-18 ani, deoarece tinerii reprezintă viitorul nostru. Dorim să invităm elevi talentați din țară și din toate colțurile lumii, care să participe la *InfoMatrix*. De aceea, dacă doriți mai multe informații despre noi și despre *InfoMatrix*, vizitați pagina noastră de web **http://www.infomatrix.ro**.

Ideea acestui concurs s-a născut ca urmare a nevoii de promovare a valorilor educaționale și de recompensare a elevilor care s-au evidențiat în acest domeniu. Timp de mai bine de 6 ani am organizat concursuri de matematică. După aceea ne-am îndreptat atenția asupra altor discipline precum fizică, geografie și limbi străine, oferindu-le elevilor talentați ocazia de a-și demonstra cunoștințele în cadrul diferitelor concursuri. Mai mult chiar, anul trecut am găzduit cea de-a 6-a ediție a *Concursului European de Proiecte Ecologice*, care a fost un factor important care ne-a făcut să extindem dimensiunea concursului *InfoMatrix*.

Concursul este organizat pe patru secțiuni, fiecare din ele având ca țintă câte o categorie de utilizatori și programatori: *desktop publishing*, *programare* pentru creatorii de *software*, *computer art* pentru pasionații de grafică și bineînțeles, *design-ul și utilizarea unui sistem de control* pentru cei interesați de controlul *hardware*.

Premiile sunt atractive atât pentru elevi, cât și pentru școlile participante. Marele premiu este un laptop (*Toshiba Satellite*) pentru cel mai bun proiect, 8 pachete *Toshiba Pocket PC* și multe periferice *Microsoft Wireless*.

Este știut că *Microsoft* este o prestigioasă companie de *software*. Suntem mândri să spunem că avem o colaborare strânsă cu *Microsoft Romania*. Aceasta a sprijinit concursul *InfoMatrix* încă de la început. În timpul fazei finale ei vor oferi seminarii despre *.NET* și tehnologia informației.

Faza finală se va desfășura pe parcursul a 5 zile. Profesori cu experiență și programatori vor susține seminarii *IT*. În ziua următoare ceremoniei decernării premiilor se vor organiza excursii opționale în diverse zone ale *României*. Punctul de atracție al fazei finale va fi expoziția de proiecte care va fi găzduită de *Palatul Național al Copiilor* în perioada 27-28 aprilie 2004. Ne dorim ca succesul acesteia să permită maturizarea concursului, transformându-l într-o competiție de tradiție, cu rezultate pe măsură și cu premii din ce în ce mai atrăgătoare.

Există tineri îndrăzneți care își pot dovedi curajul, așa că vă invităm să vizitați standurile noastre, începând din 27 aprilie la ora 10:00 la *Palatul Național al Copiilor* pentru a vă convinge. Nu uitați! Înscrierile pentru participarea la concursul internațional de anul viitor va începe la scurt timp după încheierea ediției de anul acesta. (Detalii la **http://www.infomatrix.ro**).

Sponsorii competiției

Microsoft
Shell
Eurest
PC MAGAZINE ROMÂNIA
Zappo
Info

Microsoft încurajează tinerii dezvoltatori

Alexandra Foamete

Dragi elevi informaticieni,

Ne face plăcere să vă anunțăm că în luna februarie am lansat programul "Parteneri pentru Educație", componentă a programului global "Partners in Learning", inițiat de Microsoft Corporation. Ne adresăm exclusiv elevilor și profesorilor din școli și licee și intenționăm să creăm o comunitate virtuală a tuturor celor care studiază și utilizează tehnologia informației. Programul își propune să faciliteze accesul la infrastructura IT și să încurajeze utilizarea tehnologiei informației în procesul de învățământ. Sperăm astfel să transformăm calculatorul într-un instrument de studiu, utilizat cu plăcere atât de elevi, cât și de profesori și de ce nu, de părinți.

Parteneri pentru Educație vă acordă o atenție specială vouă, elevilor pasionați de informatică. Vrem să încurajăm pasiunea voastră pentru programare, suport tehnic, aplicații etc. Suntem alături de voi și vrem să vă ajutăm să vă valorificați potențialul intelectual și creativ.

Vă invităm să aflați mai multe despre "Parteneri pentru Educație" accesând **www.microsoft.com/Romania** *și să ne scrieți despre așteptările voastre legate de acest program. Primii 50 elevi care ne vor scrie până la data de 15 aprilie 2004, răspunzând și la întrebarea: Care este viziunea Microsoft pentru educație?, vor primi cadou o șapcă inscripționată cu sigla Microsoft. Așteptăm cu drag toate scrisorile voastre pe adresa* **roacad@microsoft.com**.

Vă urez succes la olimpiadă și la concursurile școlare!

Alexandra Foamete
Education Program Manager
Microsoft România

Microsoft®

19-21 februarie, Slatina

Concursul INTERJUDEȚEAN de programare INFO-OLTENIA

Ediția din acest an a concursului Info-Oltenia s-a desfășurat în perioada 19-21 februarie la Colegiul "Radu Greceanu" din Slatina. Vă prezentăm în continuare enunțurile problemelor de la concursul de programare individual și de la cel pe echipe.

P030401: Expresie

clasa a IX-a, concursul individual

Se consideră o expresie formată din operanzi constituiți din litere mici ale alfabetului englezesc și operatori.

Operatorii folosiți sunt : * (înmulțire), / (împărțire), + (adunare binară), - (scădere binară).

Operanzii reprezintă numere raționale și sunt dați ca perechi numărător-numitor.

Expresia este corectă și nu conține paranteze.

Să se scrie un program care găsește numărul rațional în formă de fracție ireductibilă rezultată prin evaluarea expresiei.

Date de intrare

Fișierul de intrare **expresie.in** conține pe prima linie un număr natural care reprezintă numărul de caractere ale expresiei. Pe cea de-a doua linie a fișierului de intrare se află expresia a cărei valoare trebuie calculată.

Restul fișierului conține valorile operanzilor, în următorul format: pe o linie se află operandul, iar pe linia următoare se află două numere întregi, separate printr-un singur spațiu, care reprezintă numărătorul, respectiv numitorul.

Date de ieșire

Fișierul de ieșire **expresie.out** va trebui să conțină pe o singură linie două numere întregi separate printr-un singur spațiu, care reprezintă numărătorul, respectiv numitorul rezultatului.

Restricții și precizări

- lungimea unei expresii este un număr natural cuprins între 1 și 255;
- numărătorii și numitorii pentru fiecare operand sunt numere întregi cuprinse între -30000 și 30000;
- se consideră egale fracțiile de forma c/d și $(-c)/(-d)$.

Exemple

expresie.in

```
7
a+b*c-b
a
2 -3
b
1 3
c
2 4
```

expresie.out

```
-5 6
```

expresie.in

```
5
b+x/y
x
10 15
y
3 -4
b
-2 -3
```

expresie.out

```
2 -9
```

Timp de execuție: 0,5 secunde/test

P030402: Multiplu

clasa a IX-a, concursul individual

Se consideră un număr natural n și o cifră nenulă b.

Se cere să se determine cel mai mic multiplu al lui n format numai din cifrele b și 0, de forma $bbb..b00..0$ sau $bbbb...b$.

Date de intrare

Fișierul de intrare **MULTIPLU.IN** conține pe prima linie numărul n și cifra b separate printr-un singur spațiu.

Date de ieșire

Fișierul de iesire **MULTPLU.OUT** va conține un singur număr care reprezintă multiplul lui n care să respecte cerințele problemei.

Restricție

- $1 \leq n \leq 99$.

Exemplu

MULTIPLU.IN

```
4 5
```

MULTIPLU.OUT

```
500
```

Timp de execuție: 0,5 secunde/test

P030403: Tetravex

clasa a X-a, concursul individual

Tetravex este un joc care se desfășoară pe o tablă de dimensiune $N \times N$ respectând regulile de mai jos.

Există N^2 piese de formă patrată, pe fiecare latură a unei piese fiind înscrisă o cifră.

Două piese pot fi așezate una lângă cealaltă dacă au aceeași cifră înscrisă pe latura comună.

Jocul constă în așezarea tuturor pieselor pe tabla de joc.

Să se scrie un program care determină modul de așezare a pieselor pe tablă.

Date de intrare

Fișierul de intrare **tetravex.in** conține pe prima linie o valoare întreagă N, reprezentând dimensiunea tablei de joc.

Fiecare dintre următoarele N^2 linii conține câte patru cifre, separate între ele prin spațiu, care reprezintă valorile înscrise pe laturile fiecărei piese în ordinea: nord, sud, est, vest.

Date de ieșire

Fișierul de ieșire **tetravex.out** trebuie să conțină numerele de la 1 la N^2 așezate pe N linii.

Cele N^2 numere reprezintă ordinea în care se pot așeza piesele pe tablă.

Fiecare linie trebuie să conțină câte N numere, separate între ele prin spațiu.

Restricții și precizări

- $1 \leq n \leq 6$;
- se garantează existența unei soluții.

Exemplu

tetravex.in

```
2
9 5 3 8
2 7 8 9
5 1 0 2
7 5 2 8
```

tetravex.out

```
2 1
4 3
```

tetravex.in

```
3
7 2 5 4
5 7 6 3
2 8 5 4
0 3 6 5
2 4 3 1
8 0 9 6
1 7 4 4
3 9 6 5
4 1 4 5
```

tetravex.out

```
5 2 6
9 1 4
7 3 8
```

Timp de execuție: 3 secunde/test

P030404: Extra

clasa a X-a, concursul individual

Doi specialiști de la *NASA* au reușit să capteze și să decodifice mai multe semnale provenite, se pare, de pe *Marte*. Fiecare dintre ei are un sistem propriu de decodificare. Ei reușesc să transforme semnalele pe care le primesc în fraze alcătuite din cuvinte formate din litere ale alfabetului englez.

Gradul de asemănare dintre cele două fraze reprezintă numărul de cuvinte ale "subfrazei" de lungime maximă. Prin "subfrază"a unei fraze date se înțelege o frază care se poate obține din fraza dată, eliminând 0 sau mai multe cuvinte și păstrând ordinea cuvintelor rămase.

Să se găsească gradul de asemănare dintre două fraze furnizate de cei doi specialiști.

Date de intrare

Fișierul de intrare **extra.in** conține pe prima linie prima frază alcatuită din mai multe cuvinte separate prin exact un spațiu.

A doua linie conține a doua frază alcatuită din mai multe cuvinte separate prin exact un spațiu. Fiecare frază se încheie cu caracterul de sfârșit de linie (<Enter>).

Date de ieșire

Fișierul de ieșire **EXTRA.OUT** va conține pe o singură linie pe care se va afla un singur număr întreg care reprezintă gradul de asemănare a celor două fraze.

Restricții și precizări

- fiecare frază este formată din cel mult 2.000 de cuvinte;
- fiecare cuvânt este format din cel mult 12 caractere.

Exemplu

extra.in

```
we are near you
we are so close to you
```

extra.out

```
3
```

Timp de execuție: 1 secundă/test

P030405: Descompunere

clasele a XI-a și a XII-a, concursul individual

Se consideră un număr întreg n.

Să se determine numărul de posibilități distincte de descompunere a numărului n în sumă de numere naturale nenule.

Date de intrare

Fișierului de intrare **DESCOMP.IN** conține pe o singură linie numărul n.

Date de ieșire

Fișierul de ieșire **DESCOMP.OUT** trebuie să conțină pe o singură linie numărul de posibilități distincte de a descompune numărul prezent în fișierul de intrare în sumă de numere naturale nenule:

Restricție

- $1 \le n \le 90$.

Exemplu

DESCOMP.IN

```
4
```

DESCOMP.OUT

```
5
```

Explicație

Rezultatul pentru exemplul prezentat este 5 deoarece numărul 4 se poate descompune în sumă de numere naturale astfel:

```
4 = 1 +1 + 1 + 1
4 = 1 +1 + 2
4 = 2 + 2
4 = 1 + 3
4 = 4
```

Timp de execuție: 1 secundă/test

P030406: XTelecom

clasele a XI-a și a XII-a, concursul individual

Compania de telecomunicații *XTelecom* dispune de n centrale și dorește interconectarea acestora, astfel încât să existe o legătură între oricare două centrale. Pentru aceasta, *XTelecom* poate crea linii telefonice între două centrale, sau, dacă este cazul, poate cumpăra antene și crea linii telefonice între antene și centrale sau între două antene.

Din cauza așezării geografice se pot crea linii telefonice doar între anumite centrale/antene și se pot cumpăra un număr de maxim k antene.

Să se scrie un program care calculează costul minim pentru realizarea interconectării.

Date de intrare

Fișierul de intrare **xtelecom.in** conține pe prima linie trei numere naturale, separate între ele prin spațiu, care reprezintă numărul de centrale n, numărul de antene k, respectiv numărul m de linii telefonice care se pot realiza.

Fiecare dintre următoarele m linii conține câte trei numere naturale separate prin spatiu, care reprezintă sursa, destinația, respectiv costul liniilor telefonice. Sursa poate fi o centrală sau o antenă, la fel și destinația.

Antenele și centralele se dau prin numărul lor. Centralele au numerele cuprinse între 1 și n, iar antenele au numerele cuprinse între $n + 1$ și $n + k$. Liniile telefonice sunt bidirecționale.

Fiecare dintre următoarele k linii conține un singur număr natural care reprezintă costul achiziționării fiecărei antene.

Date de ieșire

Fișierul de ieșire **xtelecom.out** trebuie să conțină un singur număr natural care reprezintă costul minim pentru interconectarea celor n centrale.

Restricții și precizări

- $1 \le n \le 500$;
- $1 \le k \le 1000$;
- $1 \le m \le 2500$;
- costurile pentru liniile telefonice și pentru antene sunt numere naturale cuprinse între 1 și 250;
- din cele k antene se pot folosi oricâte;
- se garantează existența unei soluții.

Exemplu

xtelecom.in

```
2 1 3
1 2 10
```

```
1 3 1
2 3 2
5
```

xtelecom.out

```
8
```

Explicație

În cazul exemplului anterior se cumpără antena 1 care are costul 5 și se creează liniile 1-3 și 2-3 care au costurile 1, respectiv 2.

Timp de execuție: 1 secundă/test

P030407: Beculețe

clasele a IX-a și a X-a, concursul pe echipe

La un concurs pentru copii s-a propus următorul joc:

Fiecare copil primește o tablă sub forma unui caroiaj cu beculețe. Pe marginea fiecărei linii și fiecărei coloane se află câte un întrerupător care schimbă starea tuturor beculețelor de pe linia sau coloana respectivă (din aprins în stins și invers).

Inițial toate beculețele sunt stinse. La fiecare pas al jocului un concurent are voie să acționeze un singur întrerupător. Câștigator va fi numit concurentul care va reuși să aprindă un număr fix de beculete, într-un număr minim de pași.

Dându-se dimensiunile tablei de joc și numărul de beculețe care trebuie aprinse, se cere să se determine numărul minim de pași necesari pentru a aprinde beculețele.

Date de intrare

Fișierul de intrare **BECULETE.IN** conține pe o singură linie trei numere naturale separate prin spațiu care reprezintă numărul n de linii al tablei de joc, numărul m de coloane al tablei de joc, respectiv numărul k de beculețe care trebuie aprinse.

Date de ieșire

Fișierul de ieșire **BECULETE.OUT** va conține o singură linie pe care se va afla numărul minim de pași necesari pentru aprinderea beculețelor.

Restricții și precizări

- $1 \le n, m \le 100$;
- dacă există mai multe soluții, trebuie generată doar una dintre ele;
- numărul poate conține cel mult 1000000 de cifre.

Exemplu

BECULETE.IN

```
5 5 10
```

BECULETE.OUT

```
2
```

Timp de execuție: 1 secundă/test

P030408: Perm

clasele a IX-a și a X-a, concursul pe echipe

Locuitorii planetei *Anitals* comunică printr-un sistem cam ciudat cu locuitorii planetei *Avoiarc* aflată la 15101983 de ani lumină depărtare.

Primii trimit *anitalsii* un semnal codificat sub forma unei permutări de lungime N, după care *avoiarcii* răspund, în mod automat cu ajutorul unui aparat, printr-un număr P care reprezintă numărul de ordine al permutării respective, relativ la relația de ordine lexicografică. Într-o zi "softul" aparatului cu care *avoiarcii* răspund la semnalele vecinilor lor se defectează din cauza unui virus. "Softul" respectiv, prelua o permutare de lungime N dintr-un fișier și scria răspunsul care urma să fie trimis în alt fișier.

Rescrieți "softul" pentru aparatul respectiv!

Date de intrare

Prima linie a fișierului de intrare **perm.in** conține numărul N care reprezintă lungimea permutării, iar a doua linie conține N numere separate prin spațiu care reprezintă permutarea.

Date de ieșire

Fișierul de ieșire **perm.out** trebuie să conțină numărul P.

Restricție

- $1 < N < 201$.

Exemplu

perm.in

```
4
2 1 4 3
```

perm.out

```
8
```

Timp de execuție: 1 secundă/test

P030409: Gigel

clasele a XI-a și a XII-a, concursul pe echipe

Cunoscutul infractor *Gigel* a reușit încă o dată să evadeze din închisoare. Pentru a scăpa de "mâna lungă a legii" acesta și-a propus să ajungă într-o altă țară și să ceară azil politic. Ca să-și îndeplinească scopul, *Gigel* trebuie să traverseze *România*. *Poliția Română* trebuie să pună la punct o acțiune sigură pentru prinderea infractorului.

După centralizarea tuturor informațiilor disponibile, echipa de informaticieni a Poliției dispune de următoarele date:

- *Gigel* intră în *România* prin orașul codificat cu nr. 1 și iese din țară prin orașul codificat cu nr. n;
- harta completă a orașelor și străzilor din țară;
- o stradă este perfect protejată de un polițist;
- există cel puțin un traseu între oricare două orașe ale țării;
- pe o stradă se poate circula în ambele sensuri.

O problemă majoră cu care se confruntă *Poliția Română* este personalul insuficient, de aceea întreaga operațiune trebuie îndeplinită cu un număr minim de polițiști. Această ultimă cerință i-a pus în dificultate pe informaticienii poliției și tocmai de aceea au apelat la ajutorul vostru pentru a găsi soluția.

Scrieți un program care să determine numărul minim de polițiști necesari pentru a-l captura pe *Gigel*, știind că un polițist poate acoperi perfect o stradă și că *Gigel* dorește să ajungă în orașul *n*.

Date de intrare

Pe prima linie a fișierului de intrare **GIGEL.IN** se află numărul de orașe din *România*. Pe următoarele linii se găsesc străzile existente între orașele țării astfel: fiecare linie conține două numere, ceea ce înseamnă că există o stradă între orașele cu numerele respective.

Date de ieșire

Fișierul **GIGEL.OUT** va conține numărul minim de polițiști necesari pentru a-l captura pe *Gigel*.

Restricție

- numărul de orașe este un număr întreg cuprins între 5 și 250.

Exemplu

GIGEL.IN

```
8
1 2
1 3
1 4
2 3
2 4
3 4
5 8
6 8
7 8
6 7
6 8
7 8
4 5
3 6
```

GIGEL.OUT

```
2
```

Timp de execuție: 1 secundă/test

P030410: Ord

clasele a XI-a și a XII-a, concursul pe echipe

Un negustor primește într-o zi *n* cutii de marfă de greutăți diferite. El are la dispozitie *n* rafturi numerotate de la 1 la *n* și doreste să așeze cutiile astfel încât pentru orice raft cu număr de ordine *p* (*p* impar), greutatea cutiei corespunzătoare să fie mai mică decât greutatea cutiei de pe raftul cu numărul de ordine $p + 1$, iar pentru orice raft cu număr de ordine *p* (*p* par) greutatea cutiei corespunzătoare să fie mai mare decât greutatea cutiei de pe raftul cu numărul de ordine $p + 1$.

Negustorul vă roagă să-i spuneți în câte moduri *K* poate așeza cele *n* cutii pe rafturi.

Date de intrare

Fișierul de intrare **ord.in** conține pe prima linie numărul întreg *n*, iar pe cea de-a doua linie *n* numere întregi, separate între ele prin spațiu, care reprezintă greutățile celor *n* cutii.

Date de ieșire

Fișierul de ieșire **ord.out** trebuie să conțină o singură linie pe care se va afla numărul *K*.

Restricții și precizări

- $0 < n < 101$;
- greutățile sunt numere întregi distincte două câte două.

Exemplu

ord.in

```
4
12 16 5 2
```

ord.out

```
5
```

Explicație

Cutiile din exemplu se pot așeza în cinci moduri:

```
5 16 2 12
2 12 5 16
2 16 5 12
12 16 2 5
5 12 2 16
```

Timp de execuție: 1 secundă/test

Despre concurs...

Ediția din acest an este cea de-a șasea a acestui concurs. Anul cesta au participat elevi de liceu din județele: *Argeș*, *Dolj*, *Gorj*, *Mehedinți*, *Olt* și *Vâlcea*.

La proba individuală, primele trei locuri au fost ocupate la clasa a IX-a de către: ***Zegheanu Răzvan*** (*Olt*), ***Bălășel Alexandru*** (*Vâlcea*) și ***Georgescu Răducu*** (*Argeș*). La clasa a X-a podiumul a fost ocupat de: ***Ciucu Mihai*** (*Dolj*), ***Șerban Cristian*** (*Argeș*) și ***Janca Andrei*** (*Mehedinți*). Cei mai bine clasați elevi din clasa a XI-a au fost: ***Ioniță Andrei*** (*Slatina*), ***Jianu Augustin*** (*Dolj*) și ***Gheorghe Andrei*** (*Slatina*); iar din clasa a XII-a: ***Nicolae Marius*** (*Slatina*), ***Tănase Nicolae*** (*Slatina*) și ***Bilică Ionuț*** (*Dolj*).

La proba pe echipe au apărut suprize, astfel nu toți cei care au obținut rezultate bune la proba individuală au reușit să obțină cu echipa unul din primele trei locuri.

28-29 februarie 2004

Faza JUDEȚEANĂ a olimpiadei de INFORMATICĂ pentru GIMNAZIU

Vă prezentăm în continuare enunțurile celor opt probleme propuse în acest an la faza județeană a olimpiadei de informatică pentru clasele V-VIII. Subiectele au fost aceleași în toate județele țării.

Clasa a V-a

P030411: Cifre

Se consideră două numere naturale *a* și *b* care pot avea cel mult 9 cifre.

Să se determine cifrele distincte, comune numerelor *a* și *b* și numărul cel mai mare format din toate cifrele lui *a* și *b*.

Date de intrare

De la tastatură se citesc numerele *a* și *b*.

Date de ieșire

Pe ecran se vor afișa pe prima linie cifrele distincte comune ale lui *a* și *b* și pe cea de-a doua linie cel mai mare număr format din toate cifrele lui *a* și *b*.

Precizare

- datele de intrare sunt corecte (nu necesită validare).

Exemplu

Intrare

```
a=2115
b=29025
```

Ieșire

```
2 5
955222110
```

Timp de execuție: 1 secundă/test

P030412: Concurs

La un concurs de matematică participă elevi din mai multe școli din diferite orașe. Pentru a se putea deosebi între ele lucrările lor, fiecare lucrare este codificată printr-un număr natural cu 3 cifre, să zicem *abc*, unde *a* (cifra sutelor) este codul orașului, *b* (cifra zecilor) este codul școlii din orașul *a*, iar *c* (cifra unităților) este codul unui elev din școala *b* din orașul *a*.

De exemplu, lucrarea cu codul 328 este lucrarea elevului cu codul 8 de la școala cu codul 2 din orașul având codul 3.

Se cunosc: un cod (al lucrării unui elev *H*, prietenul nostru), numărul *n* de lucrări premiate și codurile fiecăreia dintre ele.

Se cere:

a) să se verifice dacă *H* este premiant sau nu.
b) să se determine numărul de premii obținute de elevii din orașul lui *H* (inclusiv *H*, dacă a fost premiat).
c) să se determine numărul de premii obținute de elevii din școala lui *H* (inclusiv *H*, dacă a fost premiat).

Date de intrare

De la tastatură se citesc: codul lui *H*, *n* și codurile celor *n* lucrări premiate.

Date de ieșire

Cele trei răspunsuri se vor afișa pe ecran, pe câte un rând.

Pentru prima cerință se va afișa un mesaj (DA sau NU), după cum *H* a obținut, sau nu un premiu. Corespunzător ultimelor două cerințe se va scrie câte un număr.

Fiecare răspuns va fi precedat de o literă (*a*, *b* sau *c*), pentru a se preciza la care dintre cerințe s-a răspuns.

Restricții și precizări

- codurile orașelor (cifra sutelor din fiecare cod) sunt cuprinse între 1 și 5;

- codurile școlilor din fiecare oraș (cifra zecilor) sunt cuprinse între 0 și 9;
- codurile elevilor din fiecare școală (cifra unităților) sunt cuprinse între 0 și 9;
- n este un număr natural nenul mai mic decât 20;
- datele de intrare sunt corecte (nu necesită validare).

Exemplu

Intrare

```
H=234
n=6
123 232 125 222 421 235
```

Ieșire

```
a) NU
b) 3
c) 2
```

Explicație

Codul lui H nu se regăsește printre cele citite (premiate), deci mesajul este NU.

Lucrările premiate din orașul lui H (cu cifra sutelor egală cu 2) sunt în număr de trei: 232, 222 și 225.

Lucrările premiate din școala lui H (cu cifra sutelor egală cu 2 și cu cifra zecilor egală cu 3) sunt în număr de două: 232 și 235.

Timp de execuție: 1 secundă/test

Clasa a VI-a

P030413: Număr de control

Gigel a primit spre păstrare un set de n cutii de greutăți nu neapărat distincte.

El a cântărit cutiile și pentru fiecare greutate distinctă a notat pe o foaie, în ordine crescătoare a greutăților, numărul de cutii cu greutatea respectivă.

Deoarece fratele său mai mic avea prostul obicei să se joace cu numerele scrise de el pe foaie, *Gigel* s-a gândit să calculeze un "număr de control" după următorul algoritm: începând de la primul număr a grupat numerele de apariții ale greutăților câte trei (dacă îi rămân numere negrupate la sfârșit, le ignoră).

Dacă într-un grup sunt numai numere pare sau numai impare notează grupul cu cifra 1, altfel îl notează cu cifra 0. Din șirul astfel obținut, se formează un număr care are ca valoare cifra zecilor egală cu numărul de valori 1 și cifra unităților egală cu numărul de valori 0, obținându-se astfel "numărul de control".

Cunoscând greutățile cutiilor, să se determine "numărul de control" și să se verifice dacă este număr prim.

Date de intrare

De la tastatură se vor citi numărul n și greutatea fiecărei cutii.

Date de ieșire

Pe ecran se va scrie pe prima linie "numărul de control", iar pe următoarea linie se va scrie 1 dacă numărul este prim, respectiv 0 în caz contrar.

Restricție

- $1 \le n \le 100$;
- fiecare greutate este un număr natural, mai mic sau egal cu 200;
- datele de intrare sunt corecte (nu necesită validare);
- se vor afișa doar două valori, fără mesaje, pe două linii diferite.

Exemplu

Intrare

```
n=21
1 3 2 6 2 6 2 8 9 8 8 9 10 8 11 18 11 12 14 15 17
```

Ieșire

```
31
1
```

Explicație

După ordonare se obține șirul: 1 2 2 2 3 6 6 8 8 8 8 9 9 10 11 11 12 14 15 17 18.

Se obține apoi: 1 3 1 2 4 2 1 2 1 1 1 1 1. Deci avem grupurile: (1 3 1), (2 4 2), (1 2 1), (1 1 1) și obținem șirul de valori: 1 1 0 1. Din valorile 1 1 0 1 se obține numărul de control 31, care este număr prim.

Timp de execuție: 1 secundă/test

P030414: Vânătoare

Vânătorul șef al regelui *Arthur* a primit însărcinare să vâneze rațele care se întorc primele din țările calde. Regele *Arthur*, fiind un tip cu idei fixe, i-a cerut vânătorului să vâneze rațele albe cu săgeți albe, iar rațele negre cu săgeți negre.

Rațele vin în rânduri (stoluri) din ce în ce mai mari: mai întâi una, apoi două, trei, cinci, opt, treisprezece, ș.a.m.d.

Se observă că numărul de rațe dintr-un rând este egal cu numărul de rațe de pe cele două rânduri anterioare. Rațele fiind niște creaturi ordonate zboară în rânduri, în care nu vei putea găsi două rațe de aceeași culoare alăturate, fiecare rând începând cu o rață albă.

Vânătorul știe că dacă a doborât o rață, atunci trebuie să le doboare pe toate de pe rândul acesteia, deoarece supraviețuitoarele le vor alerta pe celelalte rațe și acestea nu se vor mai întoarce niciodată, iar vânătorul nostru își va pierde slujba.

Știind că vânătorul a primit *ka* săgeți albe și *kb* săgeți negre, trebuie să determinați câte rânduri de rațe a doborât și câte săgeți de fiecare tip i-au rămas, știind că el vrea să-și păstreze slujba.

Date de intrare
De la tastatură se vor citi numerele *ka* și *kb* în această ordine.

Date de ieșire
Pe ecran se va afișa: numărul de rânduri de rațe doborâte, numărul de săgeți albe rămase și numărul de săgeți negre rămase.

Restricții și precizări
- $1 \le ka, kb \le 2.000.000.000$;
- datele de intrare sunt corecte (nu necesită validare).

Exemplu

Intrare	*Ieșire*
`ka=9`	`4`
`kb=10`	`2`
	`6`

Explicație
Pentru exemplul anterior avem rândurile de rațe:

```
A
A N
A N A
A N A N A
```

unde litera `A` reprezintă o rață albă, iar litera `N` reprezintă o rață neagră.

Timp de execuție: 1 secundă/test

Clasa a VII-a

P030415: Șiruri
Se consideră un vector unidimensional x cu n componente numere naturale distincte, cel mult egale cu 32 000.

Scrieți un program care să construiască vectorul y cu elemente din mulțimea $\{1, 2, ..., n\}$ astfel încât oricare ar fi numerele naturale i, j cu proprietatea că $1 \le i \le n$, $1 \le j \le n$ și $x[i] < x[j]$ să avem $y[i] < y[j]$.

Date de intrare
Fișierul de intrare **`SIRURI.IN`** va conține pe prima linie un singur număr natural care reprezintă valoarea lui n.

Cea de-a doua linie conține n numere naturale, separate între ele prin spațiu, care reprezintă elementele vectorului x.

Date de ieșire
Fișierul de ieșire **`SIRURI.OUT`** va conține o singură linie pe care se vor afla componentele vectorului y separate printr-un spațiu.

Restricții și precizări
- datele de intrare sunt corecte (nu necesită validare);
- numărul de componente ale vectorului x (n) este un întreg cuprins între 1 și 100;
- componentele vectorului x sunt numere naturale cel mult egale cu 32 000.

Exemplu
`SIRURI.IN`
```
6
12 3 7 16 10 1
```

`SIRURI.OUT`
```
5 2 3 6 4 1
```

Timp de execuție: 0,1 secunde/test

P030416: Nu te supăra, frate!
Jocul "Nu te supăra, frate!" se joacă de către doi jucători. Pe o pistă circulară cu n căsuțe numerotate de la 1 la n, în ordinea dată de sensul acelor de ceasornic, se află înscrise valorile 0, 1 și 10.

Cei doi jucători vor avea fiecare câte un pion și vor porni pe rând, din căsuța 1.

Jocul este început de jucătorul cu numărul 1.

Fiecare jucător va muta pionul său conform cu valoarea obținută prin aruncarea zarului, acumulând sau pierzând puncte în funcție de căsuța în care ajunge.

Când îi vine rândul, fiecare jucător citește valoarea zarului.

Jocul are următoarele reguli:

Câștigătorul poate fi jucătorul care ajunge primul din nou în căsuța cu numărul 1, indiferent de punctaj (exceptând cazul în care are punctaj 0);

În cazul în care șirul aruncărilor cu zarul se termină, înseamnă că jucătorii s-au plictisit și câștigă cel care a acumulat mai multe puncte, iar dacă au punctaje egale câștigă cel care se află în căsuța care are numărul de ordine mai mare.

După aruncarea zarului, jucătorul mută pionul, avansând cu atâtea căsuțe cu câte indică valoarea zarului, în ordinea acelor de ceasornic, începând numărătoarea cu căsuța următoare poziției pe care se află. Prima căsuță nu conține valoarea 0 (zero).

După mutare pot apărea următoarele situații:
- ajunge într-o căsuță cu valoarea 0 (zero) - jucătorul este penalizat, pierde toate punctele acumulate și reia jocul din poziția 1;
- ajunge într-o căsuță cu valoarea 10 - primește un bonus de 10 puncte;
- ajunge într-o căsuță cu valoarea 1 - primește 1 punct;
- ajunge într-o căsuță în care se află celălalt pion (cu excepția căsuței 1, când câștigă) - jucătorul care ajunge ultimul este penalizat, pierde toate punctele și reia jocul de la căsuța 1.

Sarcina voastră este să determinați jucătorul câștigător, pozițiile fiecărui jucător pe cerc și punctajul fiecărui jucător.

Date de intrare

Fișierul de intrare **JOC.IN** conține pe prima linie numărul n al căsuțelor din cerc.

Cea de-a doua linie conține o succesiune de n valori (0, 1 sau 10), separate printr-un spațiu, reprezentând valoarea fiecărei căsuțe.

Cea de-a treia linie conține numărul de aruncări cu zarul.

Cea de-a patra linie conține o succesiune de valori întregi cuprinse între 1 și 6, separate printr-un spațiu, reprezentând aruncarea cu zarul.

Date de ieșire

Fișierul de ieșire **JOC.OUT** va trebui să conțină pe prima linie numărul jucătorului câștigător.

Cea de-a doua linie trebuie să conțină poziția și punctajul jucătorului cu numărul 1.

Cea de-a treia linie trebuie să conțină poziția și punctajul jucătorului cu numărul 2.

Restricții și precizări

- $7 \le n \le 100$;
- datele de intrare sunt corecte (nu necesită validare).

Exemple

JOC.IN

```
10
1 1 1 1 1 10 0 1 1 0
8
3 6 2 4 1 3 5 3
```

JOC.OUT

```
2
6 10
1 3
```

JOC.IN

```
7
1 1 0 10 1 10 1
16
6 4 2 4 6 2 1 3 5 6 3 3 2 1 4 5
```

JOC.OUT

```
2
9 11
10 11
```

Timp de execuție: 0,1 secunde/test

Clasa a VIII-a

P030417: Spre culmi

Dornic de o condiție fizică perfectă, un viitor olimpic național la informatică își propune să escaladeze cea mai înaltă culme a unui un masiv muntos. Se echipează corespunzător, își cumpără un termos, îl umple cu apă, culege informațiile despre traseele existente și completează astfel fișierul de intrare **POPAS.IN**. Pe parcursul fiecărui traseu există mai multe izvoare de la care drumețul își poate umple termosul.

Știind că pe munte este bine să mergi cu pas constant și fără ruperi de ritm, își propune să atingă culmea facând cât mai puține popasuri (pentru umplerea termosului).

Dintre toate traseele existente către culme determinați-l pe cel pentru care numărul total de popasuri este minim. Dacă sunt mai multe astfel de trasee, se va alege cel care este scris ultimul în fișierul de intrare.

Date de intrare

Fișierul de intrare **POPAS.IN** conține pe prima linie un singur număr natural k, reprezentând numărul total de trasee către culme.

Fiecare dintre următoarele k linii conține descrierea câte unui traseu care constă din mai multe numere, separate prin câte un spațiu. Primul număr din descrierea unui traseu reprezintă numărul asociat traseului, fiecare traseu fiind identificat în mod unic printr-un număr natural cuprins între 1 și k. Cel de-al doilea număr reprezintă numărul r al izvoarelor cu apă rece de pe traseu, iar în continuare se află r numere, d_1, d_2, ..., d_r, care reprezintă distanța de la punctul de plecare până la fiecare izvor.

Penultima linie din fișier conține un singur număr natural t, care reprezintă distanța pentru care drumețului îi este suficientă apa din termos, iar ultima linie din fișier reprezintă distanța pe care drumețul o poate străbate fără apă.

Date de ieșire

Fișierul de ieșire **POPAS.OUT** va conține pe aceeași linie, despărțite prin spațiu, două numere care reprezintă numărul minim de popasuri necesare deplasarii, respectiv numărul traseului ales.

Dacă problema nu are soluție fișierul de ieșire va conține cifra 0.

Restricții și precizări

- $0 < k \le 100$;

- $0 < r \leq 20$;
- $0 < d_i \leq 360$, $\forall 1 \leq i \leq r$;
- $1 \leq t \leq 10$;
- $1 \leq u \leq 5$;
- datele de intrare sunt corecte (nu necesită validare).

Exemple

POPAS.IN

```
3
2 3 12 5 9
1 4 2 9 7 11
3 5 2 16 7 9 8
6
2
```

POPAS.OUT

```
1 1
```

POPAS.IN

```
2
1 3 12 5 9
2 3 2 7 11
1
2
```

POPAS.OUT

```
0
```

Timp de execuție: 1 secundă/test

P030418: Ronul fermecat

Lidorienii și *senopictii* sunt în conflict pentru *ronul fermecat*, fiind arbitrați de orintieni, aleși de părțile beligerante drept judecători. *Orintia* a propus ca:

ronul fermecat să fie ascuns printre alți k *roni* cu același aspect, dar toți realizați dintr-un material mai greu decât originalul, având o masă standard, diferită de cea a ronului femecat.

Pentru a-l descoperi, vă gândiți că aveți la dispoziție o balanță și toți cei $k + 1$ *roni*. *Lidorienii*, apoi *senopictii* vor spune un singur număr, reprezentând numărul maxim de cântăriri admis pentru descoperirea ronului fermecat.

Dacă nici una dintre părți nu descoperă numărul corect, atunci *ronul fermecat* va rămâne în *Orintia*. Dacă ambele părți descoperă numărul corect, ronul va rămâne tot la *orintieni*.

Sarcina voastră este să indicați țara care câștigă *ronul fermecat*: *Lidoria*, *Senopictia* sau *Orintia*, identificate prin literele: *L*, *S*, respectiv *O*.

Date de intrare

Fișierul de intrare **RON.IN** conține pe prima linie numărul k, iar cea de-a doua linie conține numerele RL și RS, separate printr-un singur spațiu, care reprezintă răspunsul *lidorienilor*, respectiv răspunsul *senopictilor*.

Date de ieșire

Fișierul de ieșire **RON.OUT** trebuie să conțină una dintre literele *L*, *S*, și *O*.

Litera *L* indică faptul că *Lidoria* a câștigat *ronul fermecat*. Litera *S* indică faptul că *Senopictia* a câștigat *ronul fermecat*, iar litera *O* indică faptul că *Orintia* a câștigat *ronul fermecat*.

Restricții și precizări

- $1 < K < 10000$;
- $1 \leq RL, RS \leq K$;
- *ron fermecat* reprezintă un cuboid gravat cu semnele fixe ale puterii;
- numărul maxim de cântăriri admis nu se obține cântărind un *ron* de mai multe ori și nici cântărind de cât mai multe ori *ronii*; cântărirea presupune să existe, pe fiecare braț al balanței, un număr egal de *roni* (1-1, 2-2, etc.)

Exemplu

RON.IN

```
7
1 3
```

RON.OUT

```
O
```

RON.IN

```
4
2 2
```

RON.OUT

```
O
```

Explicație

În cadrul primului exemplu, numărul maxim de cântăriri admis este 2, deci *ronul fermecat* rămâne în *Orintia*.

În cadrul celui de-al doilea exemplu, numărul maxim de cântăriri admis este 2, dar fiind egalitate, *ronul fermecat* rămâne în *Orintia*.

Timp de execuție: 1 secundă/test

Despre OJI...

La ediția din aceast an a *Olimpiadei Județene de Informatică* au participat elevi din majoritatea școlilor generale și liceelor cu profil informatică.

În urma relatărilor participanților la acest concu... dacția *GInfo* a constatat că foarte mulți co... *distrați* și nu au înțeles bine enunț... astfel au considerat că ele sunt *com...*

O altă categorie de participanți a... șeli ca în alți ani și au fost dezamăgiți că... tajul pe care l-au obținut deși, în urma r... gur caracter din program ar fi obținut p...

În general problemele au fost accesib... obținându-se și punctaje maxime la toate.

28-29 februarie 2004

Faza JUDEȚEANĂ a olimpiadei de INFORMATICĂ pentru LICEE

În perioada 28 - 29 februarie a avut loc faza județeană a olimpiadei de informatică. Subiectele propuse spre rezolvare au fost aceleași în întreaga țară. Vă prezentăm în continuare enunțurile celor șase probleme propuse spre rezolvare elevilor de liceu.

Clasa a IX-a

P030419: Expresie

Se consideră un șir format din n numere naturale nenule $x_1, x_2, \ldots, x_n$ și un număr natural m.

Să se verifice dacă valoarea expresiei $\sqrt[m]{x_1 x_2 \ldots x_n}$ este un număr natural. În caz afirmativ să se afișeze acest număr descompus în factori primi.

Date de intrare

Fișierul de intrare **exp.in** conține pe prima linie un număr natural care reprezintă valoarea lui m, iar pe cea de-a doua linie conține un număr natural care reprezintă valoarea lui n.

Cea de-a treia linie a fișierului de intrare conține n numere naturale $x_1, x_2, \ldots, x_n$ separate între ele prin câte un spațiu.

Date de ieșire

Fișierul de ieșire **exp.out** trebuie să conțină pe prima linie cifra 0, dacă valoarea expresiei $\sqrt[m]{x_1 x_2 \ldots x_n}$ nu este un număr natural, respectiv 1 dacă este un număr natural.

Dacă valoarea expresiei este un număr natural, pe fiecare dintre următoarele linii se va scrie câte o pereche formată din două numere naturale p și e, separate între ele printr-un singur spațiu, cu semnificația: *p este factor prim care apare descompunerea expresiei din enunț la puterea e ($e \geq 1$).*

Perechile se vor scrie în fișierul de ieșire în ordinea creș- a factorilor primi.

ii

5000;

0000, $i \in \{1, 2, \ldots, n\}$;

Exemple

exp.in	exp.out
2 4 32 81 100 19	0

exp.in	exp.out
2 4 32 81 100 18	1 2 4 3 3 5 1

Timp de execuție: 1 secundă/test

P030420: Reactivi

Într-un laborator de analize chimice se utilizează N reactivi. Se știe că, pentru a evita accidentele sau deprecierea reactivilor, aceștia trebuie să fie stocați în condiții de mediu speciale. Mai exact, pentru fiecare reactiv x, se precizează intervalul de temperatură $[min_x, max_x]$ în care trebuie să se încadreze temperatura de stocare a acestuia.

Reactivii vor fi plasați în frigidere. Orice frigider are un dispozitiv cu ajutorul căruia putem stabili temperatura (reprezentată de un număr întreg de grade *Celsius*) care va fi în interiorul acelui frigider .

Scrieți un program care să determine numărul minim de frigidere necesare pentru stocarea reactivilor chimici.

Date de intrare

Fișierul de intrare **react.in** conține pe prima linie numărul natural N, care reprezintă numărul de reactivi. Pe fiecare dintre următoarele N linii se află două numere întregi separate printr-un singur spațiu.

Numerele de pe linia $x + 1$ a fișierului de intrare reprezintă temperatura minimă min_x, respectiv temperatura maximă max_x de stocare a reactivului x.

Date de ieșire

Fișierul de ieșire **react.out** va conține o singură linie pe care este scris numărul minim de frigidere necesare.

Restricții și precizări

- $1 \le N \le 8000$;
- min_x și max_x sunt numere întregi cuprinse între -100 și 100, reprezentând grade *Celsius*, $\forall 1 \le x \le N$;
- $min_x \le max_x$;
- un frigider poate conține un număr nelimitat de reactivi.

Exemple

react.in
```
3
-10 10
-2 5
20 50
```

react.out
```
2
```

react.in
```
4
2 5
5 7
10 20
30 40
```

react.out
```
3
```

react.in
```
5
-10 10
10 12
-20 10
7 10
7 8
```

react.out
```
2
```

Timp de execuție: 1 secundă/test

Clasa a X-a

P030421: Perle

Granița nu se trece ușor. Asta pentru că *Balaurul Arhirel* (mare pasionat de informatică) nu lasă pe nimeni să treacă decât după ce răspunde la niște întrebări...

În acea țară există trei tipuri de perle normale (le vom nota cu 1, 2 și 3) și trei tipuri de perle magice (le vom nota cu *A*, *B* și *C*). Perlele magice sunt deosebite prin faptul că se pot transforma în alte perle (una sau mai multe, normale sau magice).

Perla magică de tipul *A* se poate transforma în orice perlă normală (una singură).

Perla magică de tipul *B* se poate transforma într-o perlă normală de tipul 2 și una magică de tipul *B*, sau într-o perlă normală de tipul 1, una magică de tipul *A*, una normală de tipul 3, una magică de tipul *A* și una magică de tipul *C*.

Perla magică de tipul *C* se poate transforma într-o perlă normală de tipul 2 sau într-o perlă normală de tipul 3, una magică de tipul *B* și una magică de tipul *C* sau într-o perlă normală de tipul 1, una normală de tipul 2 și una magică de tipul *A*.

Ca să rezumăm cele de mai sus putem scrie:

- *A* -> 1 | 2 | 3
- *B* -> 2*B* | 1*A*3*AC*
- *C* -> 2 | 3*BC* | 12*A*

Balaurul Arhirel ne lasă la început să ne alegem o perlă magică (una singură), iar apoi folosind numai transformările de mai sus trebuie să obținem un anumit șir de perle normale. Când o perlă magică se transformă, perlele din stânga și din dreapta ei rămân la fel (și în aceeași ordine). De asemenea, ordinea perlelor rezultate din transformare este chiar cea prezentată mai sus.

De exemplu, dacă balaurul ne cere să obținem șirul de perle 21132123, putem alege o perlă magică de tipul *B* și următorul șir de transformări: *B* -> 2*B* -> 21*A*3*AC* -> 21*A*3*A*12*A* -> 21132123.

Întrucât *Balaurul* nu are prea multă răbdare, el nu ne cere decât să spunem dacă se poate sau nu obține șirul respectiv de perle.

Să se determine pentru fiecare șir din fișierul de intrare dacă se poate obține prin transformările de mai sus sau nu (alegând orice primă perlă magică, la fiecare șir).

Date de intrare

Fișierul de intrare **perle.in** conține pe prima linie un număr natural *N*, reprezentând numărul de șiruri din fișierul de intrare. Fiecare a *i*-a linie dintre următoarele *N* linii descrie șirul *i*.

Un șir este dat printr-o succesiune de numere naturale despărțite prin câte un spațiu. Primul număr reprezintă lungimea șirului (L_i), iar următoarele L_i numere sunt tipurile de perle normale, în ordine, de la stânga la dreapta.

Date de ieșire

Fișierul **perle.out** trebuie să conțină *N* linii. Pe linia *i* se va scrie un singur număr 1 sau 0, 1 dacă se poate obține șirul respectiv (al *i*-lea) și 0 dacă nu se poate.

Restricții și precizări

- $0 < N < 11$;
- $0 < L_i < 10001$, $\forall 1 \le i \le N$.

Exemplu

perle.in
```
3
8 2 1 1 3 2 1 2 3
2 2 2
1 3
```

perle.out
```
1
0
1
```

Timp de execuție: 1 secundă/test

P030422: Romeo și Julieta

În ultima ecranizare a celebrei piese shakespeariene, *Romeo* și *Julieta* trăiesc într-un oraș modern, comunică prin *e-mail* și chiar învață să programeze. Într-o secvență tulburătoare sunt prezentate frământările interioare ale celor doi eroi încercând fără succes să scrie un program care să determine un punct optim de întâlnire.

Ei au analizat harta orașului și au reprezentat-o sub forma unei matrice cu n linii și m coloane, în matrice fiind marcate cu spațiu zonele prin care se poate trece (străzi lipsite de pericole) și cu X zonele prin care nu se poate trece.

De asemenea, în matrice au marcat cu R locul în care se află locuința lui *Romeo*, iar cu J locul în care se află locuința *Julietei*.

Ei se pot deplasa numai prin zonele care sunt marcate cu spațiu, din poziția curentă în oricare dintre cele 8 poziții învecinate (pe orizontală, verticală sau diagonale).

Cum lui *Romeo* nu îi place să aștepte și nici să se lase așteptat n-ar fi tocmai bine, ei au hotărât că trebuie să aleagă un punct de întâlnire în care atât *Romeo*, cât și *Julieta* să poată ajunge în același timp, plecând de acasă. Fiindcă la întâlniri amândoi vin într-un suflet, ei estimează timpul necesar pentru a ajunge la întâlnire prin numărul de elemente din matrice care constituie drumul cel mai scurt de acasă până la punctul de întâlnire. Cum probabil există mai multe puncte de întâlnire posibile, ei vor să îl aleagă pe cel în care timpul necesar pentru a ajunge la punctul de întâlnire este minim.

Scrieți un program care să determine o poziție pe hartă la care *Romeo* și *Julieta* pot să ajungă în același timp. Dacă există mai multe soluții, programul trebuie să determine o soluție pentru care timpul este minim.

Date de intrare

Fișierul de intrare **rj.in** conține pe prima linie două numere naturale, N și M, separate printr-un singur spațiu, care reprezintă numărul de linii și respectiv de coloane ale matricei, separate prin spațiu.

Pe fiecare dintre următoarele N linii se află M caractere (care pot fi doar R, J, X sau spațiu) reprezentând matricea.

Date de ieșire

Fisierul de ieșire **rj.out** va conține o singură linie pe care
eparate prin câte un spațiu

Romeo, respectiv

întâlnire și y re-

otate începând cu 1;
na soluție.

Exemplu

rj.in

```
5 8
XXR   XXX
 X  X  X
J X X  X
    XX
XXX XXXX
```

rj.out

```
4 4 4
```

Explicație

Traseul lui *Romeo* poate fi: (1,3), (2,4), (3,4), (4,4). Deci timpul necesar lui *Romeo* pentru a ajunge de acasă la punctul de întâlnire este 4. Traseul *Julietei* poate fi: (3,1), (4,2), (4,3), (4,4). Timpul necesar *Julietei* pentru a ajunge de acasă la punctul de întâlnire este de asemenea 4.

În plus (4,4), este punctul cel mai apropiat de ei cu această proprietate.

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P030423: Moșia lui Păcală

Păcală a primit, așa cum era învoiala, un petec de teren de pe moșia boierului. Terenul este împrejmuit complet cu segmente drepte de gard care se sprijină la ambele capete de câte un par zdravăn.

La o nouă prinsoare, *Păcală* iese iar în câștig și primește dreptul să strămute niște pari, unul câte unul, cum i-o fi voia, astfel încât să-și extindă suprafața de teren. Dar învoiala prevede că fiecare par poate fi mutat în orice direcție, dar nu pe o distanță mai mare decât o valoare dată (scrisă pe fiecare par) și fiecare segment de gard, fiind cam șubred, poate fi rotit și prelungit de la un singur capăt, celălalt rămânând nemișcat.

Cunoscând pozițiile inițiale ale parilor și valoarea înscrisă pe fiecare par, se cere suprafața maximă cu care poate să-și extindă *Păcală* proprietatea. Se știe că parii sunt dați într-o ordine oarecare, pozițiile lor inițiale sunt date prin numere întregi de cel mult 3 cifre, distanțele pe care fiecare par poate fi deplasat sunt numere naturale strict pozitive și figura formată de terenul inițial este un poligon neconcav.

Date de intrare

Fișierul **MOSIA.IN** conține pe prima linie un număr natural n care reprezintă numărul de pari. Fiecare a i-a linie dintre următoarele n linii conține trei numere naturale, separate printr-un singur spațiu, x_i, y_i și d_i, care au semnificația: *parul i are coordonatele inițiale (x_i, y_i) și poate fi deplasat pe distanța d_i.*

Date de ieșire

În fișierul **MOSIA.OUT** se scrie un singur număr real cu 4 zecimale care reprezintă suprafața maximă cu care se poate mări moșia.

Restricții și precizări

- $3 < N \leq 200$;
- x_i și y_i sunt numere întregi cuprinse între -1000 și 1000, $\forall 1 \leq i \leq N$;
- d_i este număr întreg cuprins între 0, exclusiv, și 20, $\forall 1 \leq i \leq N$;
- poligonul neconcav se definește ca un poligon convex cu unele vârfuri coliniare;
- pozițiile parilor sunt date într-o ordine oarecare;
- poligonul obținut după mutarea parilor poate fi concav;
- pozițiile finale ale parilor nu sunt în mod obligatoriu numere naturale.

Exemplu

MOSIA.IN	MOSIA.OUT
4	30.0000
-3 0 2	
3 0 3	
0 6 2	
0 -6 6	

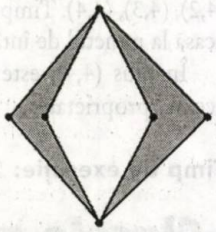


Figura 1

Explicație

Prin mutarea parilor 1 și 2 cu câte 2, respectiv 3 unități, se obține un teren având suprafața cu 30 de unități mai mare decât terenul inițial.

Timp de execuție: 1 secundă/test

P030424: Lanterna

Un agent secret are o hartă pe care sunt marcate N obiective militare. El se află, inițial, lângă obiectivul numerotat cu 1 (baza militară proprie) și trebuie să ajungă la obiectivul numerotat cu N (baza militară inamică). În acest scop, el va folosi drumurile existente, fiecare drum legând două obiective distincte.

Fiind o misiune secretă, deplasarea agentului va avea loc noaptea; de aceea, el are nevoie de o lanternă. Pentru aceasta, el are de ales între K tipuri de lanterne - o lanternă de tipul W ($1 \leq W \leq K$) are baterii care permit consumul a W wați, după consumul acestor wați, lanterna nu mai luminează.

Din fericire, unele dintre obiective sunt baze militare prietene, astfel că, o dată ajuns acolo, el își poate reîncărca bateriile complet. Agentul trebuie sa aibă grijă ca, înainte de a merge pe un drum între două obiective, cantitatea de wați pe care o mai poate consuma să fie mai mare sau egală cu cantitatea de wați pe care o va consuma pe drumul respectiv.

Cunoscând drumurile dintre obiective și, pentru fiecare drum, durata necesară parcurgerii drumului și numărul de wați consumați de lanternă, determinați tipul de lanternă cu numărul cel mai mic, astfel încât durata deplasării să fie minimă (dintre toate tipurile de lanternă cu care se poate ajunge în cel mai scurt timp la destinație, interesează lanterna cu consumul cel mai mic).

Date de intrare

Pe prima linie a fișierului **lanterna.in** se află numerele întregi N și K, separate printr-un spațiu. Pe următoarea linie se află N numere întregi din mulțimea {0, 1}. Dacă al i-lea număr este 1, aceasta înseamnă că obiectivul cu numărul i este o bază militară prietenă (adică agentul își poate reîncărca bateriile lanternei dacă ajunge la acest obiectiv); dacă numărul este 0, agentul nu își va putea reîncărca bateriile. Primul număr din linie este 1, iar ultimul este 0. Pe cea de-a treia linie a fișierului se află numărul M de drumuri dintre obiective. Fiecare din următoarele M linii conține câte patru numere întregi separate prin spații: a b T W, având semnificația că există un drum bidirecțional între obiectivele a și b ($a \neq b$), care poate fi parcurs într-un timp T și cu un consum de W wați.

Date de ieșire

În fișierul **lanterna.out** se vor afișa două numere întregi, separate printr-un spațiu: *Tmin* și *Wmin*. *Tmin* reprezintă durata minimă posibilă a deplasării de la obiectivul 1 la obiectivul N, iar *Wmin* reprezintă tipul de lanternă cu numărul cel mai mic pentru care se obține acest timp.

Restricții și precizări

- $2 \leq N \leq 50$:
- $1 \leq K \leq 1000$;
- $1 \leq M \leq N \cdot (N-1)/2$;
- între două orașe diferite poate exista maximum un drum direct;
- pentru fiecare drum, durata parcurgerii este un număr întreg cuprins între 1 și 100, iar numărul de wați consumați este un număr întreg între 0 și 1000;
- se garantează că există cel puțin un tip de lanternă pentru care deplasarea să fie posibilă;
- punctajul pentru un test se va acorda în felul următor:
  - 30%: dacă este determinat corect *Tmin*;
  - 100%: dacă sunt determinate corect atât *Tmin*, cât și *Wmin*.

Exemplu

lanterna.in

```
7 10
1 0 1 0 0 0 0
7
1 2 10 3
1 4 5 5
2 3 10 3
4 3 15 1
3 6 4 3
6 5 2 2
5 7 1 0
```

lanterna.out

```
27 6
```

Timp de execuție

Rundele #16 - #25

În cadrul acestui articol vom publica detalii despre desfășurarea rundelor #16-#25 ale ediției 2003/2004 a concursului de programare Bursele Agora, organizat de revista noastră.

P020424: Tăiere

Problema se rezolvă relativ ușor folosind o strategie *Greedy*. La fiecare pas vom încerca să eliminăm un grup astfel încât numărul rămas să fie cât mai mic posibil.

Grupul ales este primul care reprezintă un număr care este mai mare decât numărul reprezentat de grupul anterior (precedentele x cartonașe), și decât numărul reprezentat de grupul următor (următoarele x cartonașe).

Pentru a ne asigura că grupurile anterior și următor există întotdeauna, vom adăuga x cartonașe fictive la începutul și la sfârșitul șirului. Pe aceste cartonașe se va scrie cifra 0.

Aparent, apare o problemă cauzată de faptul că după eliminarea a x cartonașe, următoarele x eliminate nu ar forma un grup în șirul inițial. Din fericire, aceasta nu este o problemă reală deoarece în total sunt eliminate $2 \cdot x$ cartonașe aflate pe poziții consecutive, deci pot fi formate două grupuri a câte x cartonașe.

Să considerăm următorul șir de cartonașe:

1 2 9 6 5 4 3 1

din care trebuie eliminate două grupuri a câte trei cartonașe.

Inițial vom adăuga câte trei zerouri la începutul și la sfârșitul șirului. Astfel obținem:

0 0 0 1 2 9 6 5 4 3 1 0 0 0

Potrivit strategiei prezentate, primul grup eliminat este 9 6 5 și după efectuarea acestei operații obținem șirul:

0 0 0 1 2 4 3 1 0 0 0

La al doilea pas vom alege grupul 2 4 3 și vom obține
l:

0 0 1 1 0 0 0

ourilor de la începutul și de la
rezultatul căutat: 11.

rimul grup eliminat se află pe
ial, iar cartonașele din al doi-
, 6 și 7 în acest șir, ceea ce nu
ilor din enunț.

nașe aflate pe pozițiile conse-
pot forma două grupuri: po-
5, 6 și 7.

Ca urmare, grupurile eliminate vor fi, de fapt, 296 și 543, dar rezultatul final este exact același: numărul obținut este 11.

Strategia prezentată permite întotdeauna obținerea celui mai mic număr posibil.

Au rezolvat corect...

Problema s-a dovedit a fi relativ ușor de rezolvat; 29 dintre cei 45 de concurenți care au participat la această rundă au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Irinel Besliu, România
Andrei Blaj, România
Iulian Blajuț, România
Adrian Cârcu, România
Liviu Ciortea, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Ștefan Gheorghe, România
Andrei Gönczi, România
Sergiu Hlihor, România
Vasanth Jai, India
Andrei Janca, România
Johannes Slotta, Germania
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Marius Nicolae, România
Mircea Pașoi, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Alexandru Peptan, România
Mihail-Cosmin Pițt-Rada, România
Alexandru Popa, România
Claudiu Rad, România
Valentin Stanciu, România
Sorin Stancu-Mara, România
Adrian Vladu, România

P020425: Sume

Această problemă poate fi rezolvată foarte ușor păstrând un șir de valori booleene care să indice dacă o anumită sumă poate sau nu fi obținută.

Datorită faptului că avem cel mult 100 de numere ale căror valori sunt cel mult 10000, suma maximă care poate fi obținută este 1000000, deci am avea nevoie de 1000001 valori. Ca urmare, va trebui să folosim un singur bit pentru reprezentarea unei sume, ceea ce reduce necesarul de memorie la 125001 de octeți pentru un șir.

Inițial vom considera că nu poate fi obținută decât suma 0. În continuare, la fiecare pas, vom lua în considerare un număr. Vom parcurge șirul sumelor obținute la pasul anterior (bit cu bit) și, pentru fiecare bit cu valoarea 1, la suma corespunzătoare vom adăuga numărul curent. Rezultatul operației va reprezenta o sumă care poate fi obținută cu siguranță, așadar pentru pasul următor va trebui să setăm la valoarea 1 bitul corespunzător noii sume.

În final vom număra biții cu valoarea 1 din întregul șir și vom scrie rezultatul.

Se observă că, dacă parcurgem șirul de la stânga la dreapta (începând cu valoarea 0), în momentul în care bitul corespunzător unei sume devine 1, dacă acesta este setat în același șir, atunci când se va ajunge la poziția respectivă nu vom putea ști dacă bitul a fost setat la pasul curent sau la un pas anterior.

Există cel puțin soluții simple care pot evita astfel de complicații. Se poate folosi un șir de biți auxiliar care va indica sumele obținute la începutul fiecărui pas și noile sume sunt indicate doar în acest șir. După încheierea execuției unui pas șirul auxiliar devine șirul curent. O a doua posibilitate este utilizarea unui șir auxiliar ale cărui valori sunt nule la începutul fiecărui pas. Din nou, noile sume sunt setate în șirul auxiliar, iar la încheierea execuției pasului șirul curent se obține printr-o simplă conjuncție logică între cele două șiruri, iar valorile tuturor biților din șirul auxiliar devin 0.

Totuși, există o soluție mult mai simplă și mai eficientă. Problema amintită anterior este evitată prin simpla parcurgere a șirului de biți de la dreapta spre stânga (dinspre suma maximă spre 0). Fiecare bit nou-setat se va afla la dreapta poziției curente, deci nu va mai fi întâlnit la pasul curent datorită deplasării spre stânga.

Au rezolvat corect...

Dintre cei 46 de concurenți care au participat la această etapă a concursului, doar șapte au reușit să obțină punctajul maxim:

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Marius Dumitran, România
Vasanth Jai, India
Mircea Pașoi, România
Mihail-Cosmin Pițˇ-Rada, România
Adrian Vladu, România

P020426: Warp

Problema se reduce la o simplă determinare a celui mai lung dintre drumurile minime dintre toate perechile de noduri ale unui graf neorientat.

Apare o singură diferență: costul unui drum este dat de produsul costurilor muchiilor componente și nu de suma lor.

Această dificultate este evitată foarte ușor prin simpla logaritmare a costurilor muchiilor (obținem astfel drumuri în care înmulțirile sunt transformate în adunări).

În final vom obține logaritmul celui mai lung dintre drumurile minime.

Valoarea drumului minim ar putea fi obținută printr-o exponențiere dar, datorită faptului că această valoare este foarte mare soluția nu este viabilă.

O posibilitate este păstrarea traseelor și reconstituirea drumului obținut. Rezultatul se obține prin înmulțirea costurilor muchiilor de pe acest drum (folosind operații cu numere mari).

Au rezolvat corect...

Doar 31 de concurenți au trimis rezolvări la această etapă; dintre aceștia, șapte au reușit să obțină punctajul maxim:

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Silviu-Ionuț Gănceanu, România
Mihai-Vlad Pantiș, România
Mircea Pașoi, România
Mihail-Cosmin Pițˇ-Rada, România
Sorin Stancu-Mara, România

P020427: Găuri

Problema poate fi foarte ușor rezolvată prin folosirea unui simplu algoritm de umplere. Cu ajutorul acestuia vom putea determina toate zonele.

După determinarea zonelor le vom elimina pe cele care nu sunt găuri (conțin cel puțin un element care se află pe prima linie, prima coloană, ultima linie sau ultima coloană) și alegem cea mai mare zonă rămasă.

Pentru simplitate, putem eticheta elementele care formează zonele folosind numere întregi distincte. După identificarea unei zone vom păstra numărul de element care o formează (dimensiunea sa) într-un vector.

Vom parcurge apoi prima linie, prima coloană, ultima linie și ultima coloană și vom determina toate etichetele care apar. Pentru fiecare etichetă determinată "colorăm" zona respectivă prin setarea d

În final vom determina elem reprezintă dimensiunile zonelo

Bineînțeles, există o mulțim tificare a zonelor și de eliminar

De exemplu, se poate por (se iau în considerare toate ele prima coloană și ultima colo umplere, se determină

atribuie valoarea 1 tuturor elementelor care fac parte din acestea.

Au rezolvat corect...

Dintre cei 29 de participanți la această rundă a concursului de programare *Bursele Agora*, zece au reușit să obțină punctajul maxim:

Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Kiril Minkov, Bulgaria
Mihai-Vlad Pantiș, România
Mircea Pașoi, România
Mihail-Cosmin Pițt-Rada, România
Claudiu Rad, România
Adrian Vladu, România

P020428: Daneel

Călătoria lui *Daneel* este foarte "asemănătoare" cu cea pe care "a efectuat-o" *Bob* în 1998 la ediția a V-a a *Olimpiadei de Informatică a Europei Centrale* care s-a desfășurat la *Zadar*, în *Croația*.

Așadar, această problemă este foarte asemănătoare cu problema *Șosele* de la concursul desfășurat în urmă cu șase ani.

Singura diferență o constituie faptul că *Daneel* poate călători pe un drum în ambele sensuri, spre deosebire de *Bob* care putea călători într-un singur sens.

Enunțul original a fost publicat în *GInfo* 8/8 (noiembrie 1998), iar soluția în *GInfo* 9/1 (ianuarie 1999). Prezentăm în continuare o versiune prescurtată a descrierii soluției.

În primul rând vom reformula problema în termeni din teoria grafurilor:

Se consideră un multigraf ale cărui vârfuri sunt identificate prin numerele 1, 2, ..., n. Între două vârfuri pot exista mai multe muchii. Fiecărei muchii i se atașează o lungime și o taxă care trebuie plătită la traversarea sa. De asemenea, este cunoscută o sumă k. Trebuie să se determine lugimea celui mai scurt drum (dacă acesta există) de la nodul 1 la nodul n pentru care taxa totală plătită nu depășește valoarea k.

Pentru fiecare muchie vom memora extremitățile, lungimea și costul.

Vom construi o listă ale cărei elemente sunt înregistrări care conțin câte trei câmpuri:

- vârful curent la care s-a ajuns (pe un drum care pornește de la vârful 1);
- taxa care a trebuit plătită pe drumul prin care s-a ajuns la nodul respectiv;
- lungimea drumului care pornește de la nodul 1 și ajunge la nodul respectiv.

Vom păstra într-un vector distanțele minime ale drumurilor de la vârful 1 la fiecare dintre celelalte vârfuri care corespund unor drumuri al căror costuri sunt cel mult egale cu *k*.

Elementele listei sunt întotdeauna ordonate în funcție de cost. În momentul în care se încearcă inserarea unui element în listă, atunci acesta nu va fi adăugat dacă în listă există un alt element care conține informații despre același nod, dar pentru care atât costul, cât și distanța sunt mai mici.

Pentru simplitate, vom ordona muchiile în funcție de una dintre extremități și vom identifica pozițiile la care încep muchiile corespunzătoare fiecărui nod. Orice muchie va fi dublată pentru a permite traversarea sa în ambele sensuri.

Rezolvarea se bazează pe o parcurgere în lățime a grafului, plecând de la lista care conține un singur element pentru care nodul curent este 1, iar distanța și costul sunt ambele 0.

La fiecare pas vom extrage primul element al listei. Elementul nu se expandează dacă distanța corespunzătoare este mai mare decât distanța minimă pentru nodul respectiv. În caz contrar se determină vârfurile în care se poate ajunge din acest nod și se încearcă adăugarea în listă a unor elemente corespunzătoare acestor vârfuri.

Se calculează valorile pentru costuri și distanțe (adăugând distanțele și costurile pentru muchiile corespunzătoare) și se inserează elemente în liste numai dacă noul cost este cel mult egal cu *k*.

Algoritmul se încheie când lista devine vidă. Rezultatul va fi dat de distanța minimă corespunzătoare nodului identificat prin *n*.

Au rezolvat corect...

La această a douăzecea etapă a concursului am primit doar 23 de rezolvări. Cinci dintre participanți au obținut punctajul maxim:

Mugurel-Ionuț Andreica, România
Silviu-Ionuț Gănceanu, România
Kiril Minkov, Bulgaria
Mircea Pașoi, România
Sorin Stancu-Mara, România

P020429: Munte

Un număr are aspect de munte dacă pe primele poziții se află cifre în ordine crescătoare, iar pe ultimele poziții se află cifre în ordine descrescătoare.

Pentru a rezolva problema vom determina, pentru fiecare cifră în parte, lungimea celui mai lung subșir crescător care se termină cu cifra respectivă, precum și lungimea celui mai lung subșir descrescător care începe în dreptul cifrei respective.

Pentru fiecare cifră vom calcula sumele celor două lungimi, iar "vârful" muntelui va fi dat de cifra pentru care suma este maximă.

Soluția problemei va fi dată de diferența dintre numărul de cifre al numărului dat și suma respectivă, la care se va adăuga valoarea 1.

Valoarea 1 trebuie adăugată deoarece "vârful" muntelui este considerat de două ori: în șirul crescător și în șirul descrescător.

Au rezolvat corect...

Dintre cei 56 de concurenți care au trimis soluții la această problemă, 37 au obținut punctajul maxim:

Marius Airinei, România
Victor Ambrus, România
Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Eduard-Gabriel Băzăvan, România
Călin Bindea, România
Andrei Blaj, România
Iulian Blajuț, România
Adrian Cârcu, România
Cristian Cernat, România
Liviu Ciortea, România
Andrei Deac, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Claudiu-Dan Gheorghe, România
Andrei-Gabriel Ion, România
Kazmenko Ivan, Federația Rusă
Vasanth Jai, India
Andrei Janca, România
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Mihai Moalfă, România
Cosmin Molea, România
Mihai-Vlad Pantiș, România
Mircea Pașoi, România
Ciprian Păduraru, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Alexandru Peptan, România
Vlad Petcu, România
Mihail-Cosmin Pițˇ-Rada, România
Victor Podeanu, România
Tudor-Alexandru Rabaea, România
Renald, Indonezia
Silviu Julean, România
Adrian Vladu, România

P020430: Puncte

În primul rând vom observa că determinantul folosit pentru calcularea ariei unui triunghi determinat de trei puncte ale căror coordonate sunt întregi trebuie să fie un număr par.

De asemenea, se observă că paritatea determinantului este determinată de paritatea celor șase coordonate.

Din acest motiv vom împărți punctele în patru categorii (ambele coordonate pare, ambele coordonate impare, prima coordonată pară, iar a doua impară și prima coordonată impară, iar a doua pară). Vom număra punctele din fiecare categorie și vom calcula pe baza unei formule simple modul în care se pot alege puncte din diferite categorii pentru a obține un determinant par.

Algoritmul este liniar deoarece toate operațiile se efectuează în timp constant, cu excepția celei de împărțire a punctelor în cele patru categorii.

Au rezolvat corect...

La această rundă au participat 55 de concurenți dintre care 24 au obținut punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Andrei Blaj, România
Adrian Cârcu, România
Cristian Cernat, România
Liviu Ciortea, România
Adrian Diaconu, România
Ioan-Dan Docleanu, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Ștefan Gheorghe, România
Kazmenko Ivan, Federația Rusă
Vasanth Jai, India
Haryanto Lego, Statele Unite ale Americii
Adina Pampu, România
Mircea Pașoi, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Alexandru Peptan, România
Mihail-Cosmin Pițˇ-Rada, România
Tudor-Alexandru Rabaea, România
Renald, Indonezia
Cristian-George Strat, România

P020431: Era imperiilor

Pentru ca aria poligonului să fie maximă, punctele care îl determină trebuie să se afle pe același cerc.

Vom determina raza acestui cerc folosind *metoda căutării binare*. Pentru fiecare rază pe care o verificăm, vom încerca să amplasăm punctele pe cerc. Dacă reușim să închidem poligonul, am găsit raza corectă. Dacă nu, vom verifica dacă ultimul punct a "trecut" (în sensul considerat) de primul punct. Dacă ultimul punct trece de primul punct, atunci raza căutată este mai mare decât cea aleasă, iar dacă nu, raza căutată este mai mică.

După determinarea razei vom calcula aria poligonului folosind o simplă triangularizare (coordonatele sunt determinate deja de la pasul căutării binare).

Mai trebuie tratat cazul în care nu se poate forma nici un poligon. În această situația, aria care poate fi împrejmuită este, evident egală cu 0.

Au rezolvat corect...
La această etapă au participat 27 de concurenți, dintre care numai cinci au obținut punctajul maxim:

Liviu Ciortea, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Daniel Păsăilă, România
Tudor-Alexandru Rabaea, România

P020432: Aladin

Se observă foarte ușor că orice configurație binară a primei linii identifică unic întregul covor cu două excepții:
- configurații de forma: `0 1 0 1 0 1 ...`
- configurații de forma: `1 0 1 0 1 0 ...`

Pentru aceste două tipuri de configurații, pentru fiecare linie avem două opțiuni (putem alege oricare dintre cele două configurații pe linia respectivă).

Ca urmare avem $2^m - 2$ configurații pe prima linie care identifică în mod unic întregul covor și alte 2^n configurații care pot fi obținute folosind pe prima linie una dintre cele două configurații amintite. În concluzie, numărul total de configurații posibile este $2^m + 2^n - 2$.

Au rezolvat corect...
Dintre cei 47 de participanți la această etapă, 28 au reușit să obțină numărul maxim de puncte.

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Alexandru Anton, România
Eduard-Gabriel Băzăvan, România
Oana Belei, România
Andrei Blaj, România
Adrian Cârcu, România
Liviu Ciortea, România
Andrei Costaș, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionut Ganceanu, România
Kazmenko Ivan, Federația Rusă
Vasanth Jai, India
Slotta Johannes, Germania
Haryanto Lego, Statele Unite ale Americii
Dalian Valeriu Moraru, România
Marius Nicolae, România
Mircea Pașoi, România
Ciprian Păduraru, România
Daniel Păsăilă, România
Alexandru Peptan, România
Mihail-Cosmin Pițt-Rada, România
Victor Podeanu, România
Calin Popa, România
Claudiu Rad, România
Renald, Indonezia
Adrian Vladu, România

P020433: Santinele

Problema se reduce la determinarea poziției unui cerc de rază dată care conține în interiorul său cât mai multe dintre punctele date.

Pentru a rezolva problema vom lua în considerare fiecare punct în care se află o santinelă și vom presupune că punctul respectiv se află pe circumferința cercului de rază r căutat.

În aceste condiții problema se reduce la a rezolva următoarea subproblemă, pentru fiecare punct.

*Se cunosc coordonatele a **n** puncte. Să se determine un cerc care conține pe circumferința sa unul dintre punctele date, are raza **r** și conține în interiorul său cât mai multe dintre punctele date.*

Pentru început vom sorta celelalte n - 1 puncte în funcție de valorile unghiurilor determinate de dreapta care unește cele două puncte și dreapta orizontală care trece prin punctul ales.

În continuare vom considera un cerc oarecare de rază r care trece prin punctul ales și numărăm punctele aflate în interiorul acestui cerc (inclusiv cele aflate pe circumferința cercului).

Cunoscând ordinea punctelor (stabilită în urma sortării anterioare) putem mișca centrul cercului determinat în sens trigonometric în jurul punctului ales, astfel încât la fiecare rotație numărul punctelor din interiorul cercului să crească sau să scadă cu 1.

După efectuarea celor n - 1 rotiri vom ști care este numărul maxim de puncte care se află în interiorul unui cerc a cărui rază este r și pe a cărui circumefrință se află un punct dat.

Aplicând același procedeu pentru toate punctele, vom putea determina numărul maxim de puncte care se află în interiorul unui cerc de rază r.

Se observă că pentru fiecare punct efectuăm o sortare al cărei ordin de complexitate este $O(n \cdot \log n)$ și un număr de $O(n)$ rotiri, ordinul de complexitate al fiecărei operații de rotire fiind $O(1)$.

Ca urmare, ordinul de complexitate al operațiilor efectuate pentru fiecare punct ales este egal cu $O(n \cdot \log n) + O(n) = O(n \cdot \log n)$.

Datorită faptului că avem în total n puncte, ordinul de complexitate al acestui algoritm de determinare a numărului maxim de puncte care se află în interiorul unui cerc a cărui rază este cunoscută este: $O(n) \cdot O(n \cdot \log n) = O(n^2 \cdot \log n)$.

A rezolvat corect...
Aceasta s-a dovedit a fi cea mai dificilă problemă propusă în prima jumătate a primei faze a concursului nostru. Un singur concurent dintre cei 34 care au trimis rezolvări a reușit să obțină punctajul maxim:

Adrian Cârcu, România

Bursele Agora 2003/2004 Top 50

1. Mugurel-Ionuț Andreica, București	2177
2. Adrian Cârcu, Bistrița	2090
3. Mircea Pașoi, Ploiești	1937
4. Silviu-Ionuț Gănceanu, Bârlad	1842
5. Chattopadhyay Amit, India	1833
6. Haryanto Lego, Statele Unite ale Americii	1770
7. Ivan Kazmenko, Federația Rusă	1760
8. Kiril Minkov, Bulgaria	1759
9. Liviu Ciortea, București	1753
10. Vasanth Jai, India	1718

11. Daniel Păsăilă, Suceava	1602
12. Mihail-C. Pițˇ-Rada, Dr. Tr. Severin	1568
13. Adrian Vladu, București	1554
14. Marius Dumitran, București	1511
15. Alexandru Peptan, Dr. Tr. Severin	1410
16. Csaba Pătcaș, Oradea	1384
17. Adrian Diaconu, București	1290
18. Mihai-Vlad Pantiș, Cluj-Napoca	1253
19. Claudiu Rad, Deva	1207
20. Sergiu Hlihor, Lugoj	1017
21. Sorin Stancu-Mara, București	997
22. Alexandru Popa, București	927
23. Tudor-Alexandru Rabaea, Constanța	913
24. Iulian Blajuț, Luncani	849
25. Andrei Blaj, Suceava	779
26. Victor Podeanu, București	750
27. Valentin Stanciu, București	722
28. Andrei Gönczi, Arad	700
29. Andrei Janca, Dr. Tr. Severin	698
30. Ioan-Dan Docleanu, Nădlac	686
31. Marius Nicolae, Slatina	673
32. Mihai Moalfă, Sîngeorz-Băi	633
33. Eduard-G. Băzăvan, Dr. Tr. Severin	631
34. Irinel Besliu,Vaslui	592
35. Dalian-Valeriu Moraru, București	569
36. Cristian Cernat, Târgu Jiu	567
37. Ciprian Păduraru, Bârlad	542
38. Andrei-Ionuț Tudor, Slobozia	509
39. Călin Bindea, Bistrița	497
40. Cosmin Molea, Bascov	450
41. Renald, Indonezia	443
42. Adrian-Romulus Ciripan, Alba-Iulia	436
43. Sorin Vultureanu	409
44. Dragoș Cavulea, Botoșani	394
45. Adina Pampu, Făgăraș	386
46. Cosmin Matei, Ploiești	380
47. Claudiu-Dan Gheorghe, Slobozia	376
47. Johannes Slotta, Germania	376
49. Andrei-Gabriel Ion, Constanța	343
50. Stelian Kelemen, Bistrița	335

Rundele #26 - #40

În continuare vă vom prezenta enunțurile celor 15 probleme propuse spre rezolvare la rundele 26-40 ale ediției din acest an a concursului de programare Bursele Agora.

P030425: Perechi

Se consideră o pereche de numere naturale a și b. Asupra perechii pot fi efectuate următoarele transformări:

- (a, b) devine $(a - b, b)$;
- (a, b) devine $(a + b, b)$;
- (a, b) devine (b, a).

Să se determine numărul minim de transformări succesive care duc la apariția perechii (c, d).

Date de intrare

Fișierul de intrare `PAIRS.IN` conține două linii. Pe prima linie se află numerele a și b care formează perechea dată, iar pe cea de-a doua linie se află numerele c și d care formează perechea care trebuie obținută. Numerele de pe o linie sunt separate prin spații.

Date de ieșire

Fișierul de ieșire `PAIRS.OUT` trebuie să conțină o singură linie pe care se va afla numărul minim de transformări succesive necesare. În cazul în care perechea (c, d) nu poate fi obținută, valoarea scrisă trebuie să fie -1.

Restricție

- $0 \le a, b, c, d \le 1000$.

Exemplu

`PAIRS.IN`

```
2 2
2 4
```

`PAIRS.OUT`

```
2
```

Timp maxim de execuție/test: 1 secundă

P030426: Triunghi

Se consideră un număr natural a.

Să se determine numărul de triunghiuri dreptunghice distincte pentru care lungimea celei mai mici laturi este a și lungimile celorlalte laturi sunt numere întregi. Se consideră că un triunghi ale cărui laturi sunt a, b și c este identic cu triunghiul ale cărui laturi sunt a, c și b.

Date de intrare

Fișierul de intrare `TRIANGLE.IN` conține pe o singură linie numărul a.

Date de ieșire

Fișierul de ieșire `TRIANGLE.OUT` trebuie să conțină o singură linie pe care se va afla numărul de triunghiuri dreptunghice distincte pentru care lungimea celei mai mici laturi este a și lungimile celorlalte laturi sunt numere întregi.

Restricție

- $3 \le a \le 2.000.000.000$.

Exemplu

`TRIANGLE.IN`

```
9
```

`TRIANGLE.OUT`

```
2
```

Timp maxim de execuție/test: 0,5 secunde

P030427: Fotbal

Ciobănașul *Ion* se plictisește păzind oile și s-a gândit să diversifice activitățile. El a început să învețe un grup de oi să joace fotbal și vrea să împartă aceste oi în două echipe și să organizeze un meci între ele, dar pentru ca meciul să fie cât mai interesant el a impus restricția ca echipele să aibă valori cât mai apropiate posibil (deoarece meciul de fotbal este între oi, nu trebuie ca numărul de oi dintr-o echipă să fie egal cu numărul de oi din cealaltă echipă, dar fiecare oaie care știe fotbal trebuie repartizată în una dintre echipe).

Ciobănașul vrea să știe câte astfel de repartizări ale oilor în două echipe există astfel încât diferența de valoare dintre echipe să fie minimă și care este acea valoare.

Date de intrare

Fișierul de intrare **SOCCER.IN** conține pe prima linie numărul n al oilor care știu să joace fotbal, iar pe a doua linie conține n numere întregi, separate între ele prin spații, care reprezintă valorile fotbalistice pentru fiecare oaie.

Date de ieșire

Fișierul de ieșire **SOCCER.OUT** trebuie să conțină o singură linie pe care se vor afla două numere întregi separate între ele printr-un singur spațiu.

Primul număr reprezintă diferența minimă care se poate obține între valorile celor două echipe, iar cel de-al doilea număr reprezintă numărul de repartizări în două echipe ale oilor astfel încât diferența dintre valorile celor două echipe să fie minimă.

Restricții și precizări

- $2 \le n \le 32$;
- valoarea fotbalistică a unei oi este un număr întreg cuprins între 1 și 1000000.

Exemplu

SOCCER.IN

```
4
1 2 4 6
```

SOCCER.OUT

```
1 2
```

Timp maxim de execuție/test: 2 secunde

P030428: Dispozitive de comunicare

Agora GSM deține n dispozitive pentru comunicație identificate prin numere cuprinse între 1 și n.

Din nefericire nu este posibilă comunicația între oricare două dispozitive. Mai mult, dacă un mesaj poate fi transmis de la un dispozitiv x la un dispozitiv y, nu este sigur că poate fi transmis și un mesaj de la dispozitivul y la dispozitivul x.

Compania dorește să determine numărul minim al grupurilor care pot fi formate astfel încât să existe posibilitatea comunicării directe sau indirecte între oricare două dispozitive care fac parte din același grup.

Date de intrare

Prima linie a fișierului de intrare **DEVICES.IN** conține numărul n al dispozitivelor de comunicare și numărul m al legăturilor care pot fi stabilite între dispozitive. Fiecare dintre următoarele m linii va conține două numere întregi x și y cu semnificația: *există posibilitatea de a transmite un mesaj de la dispozitivul x la dispozitivul y*. Numerele de pe o linie vor fi separate prin spații.

Date de ieșire

Fișierul de ieșire **DEVICES.OUT** trebuie să conțină o singură linie pe care se va afla numărul minim al grupurilor care pot fi formate astfel încât să existe posibilitatea comunicării directe sau indirecte între oricare două dispozitive care fac parte din același grup.

Restricții și precizări

- $1 \le n \le 500$;
- $1 \le m \le 5000$;
- există posibilitatea ca un grup să fie format dintr-un singur dispozitiv;
- va exista întotdeauna cel puțin o gaură.

Exemplu

DEVICES.IN

```
4 5
1 2
2 3
2 4
3 1
3 4
```

DEVICES.OUT

```
2
```

Timp maxim de execuție/test: 2 secunde

P030429: Media aritmetică

Se consideră un șir a format din n numere întregi. Să se determine numărul k al valorilor a_i cu proprietatea că a_i este media aritmetică a celorlalte n - 1 numere din șir.

Date de intrare

Prima linie a fișierului de intrare **AVERAGE.IN** conține valoarea n.

Cea de-a doua linie conține n numere întregi, separate prin spații, care reprezintă valorile elementelor din șirul a.

Date de ieșire

Fișierul de ieșire **AVERAGE.OUT** trebuie să conțină pe prima linie numărul k determinat.

Cea de-a doua linie trebuie să conțină cele k poziții ale numerelor a_i care au proprietatea cerută. Acestea vor fi separate prin spații.

În cazul în care avem $k = 0$, cea de-a doua linie a fișierului de ieșire va fi vidă.

Restricții și precizări

- $1 \le n \le 1000000$;
- $1 \le a_i \le 1000$, $\forall 1 \le i \le n$;
- primul element al șirului a se află pe poziția 1.

Exemplu

AVERAGE.IN

```
5
1 2 3 4 5
```

AVERAGE.OUT

```
1
3
```

Timp maxim de execuție/test: 1 secundă

P030430: Keops

Doi hoți au reușit să intre în *Marea Piramidă* a lui *Keops*. Nici unul dintre ei nu știe de prezența celuilalt. Piramida poate fi privită ca un caroiaj cu m linii și n coloane.

Se cunosc coordonatele la care se află inițial hoții. În fiecare minut un hoț se poate deplasa din poziția sa într-o poziție vecină pe verticală sau pe orizontală dacă aceasta este accesibilă.

Va trebui să determinați numărul minim de minute necesare pentru ca cei doi hoți să se întâlnească.

Date de intrare

Prima linie a fișierului de intrare **KEOPS.IN** conține valorile m și n care reprezintă dimensiunile piramidei. Cea de-a doua linie conține coordonatele l_1 și c_1 (linia și coloana) ale primului hoț, iar cea de-a treia linie conține coordonatele l_2 și c_2 (linia și coloana) ale celui de-al doilea hoț.

Numerele de pe aceste linii sunt separate prin spații.

Pe fiecare dintre următoarele m linii se află un șir de n numere neseparate prin spații. Valorile acestor numere pot fi 0 sau 1.

Valoarea 0 indică faptul că celula corespunzătoare este accesibilă, iar valoarea 1 indică faptul că celula este inaccesibilă.

Date de ieșire

Fișierul de ieșire **KEOPS.OUT** trebuie să conțină o singură linie pe care se va afla numărul minim de minute necesare pentru ca cei doi hoți să se întâlnească.

Dacă nu există posibilitatea ca cei doi hoți să se întâlnească, valoarea scrisă va fi -1.

Restricții și precizări

- $2 \le m, n \le 250$;
- liniile sunt numerotate de la 1 la m;
- coloanele sunt numerotate de la 1 la n;
- hoții nu se pot afla în aceeași celulă la început;
- ambii hoți se află inițial în celule accesibile;
- hoții nu vor părăsi piramida.

Exemplu

KEOPS.IN

```
5 5
5 5
1 1
00000
11110
00000
01111
00000
```

KEOPS.OUT

```
8
```

Timp maxim de execuție/test: 1 secundă

P030431: Colectare

Anual, regele *Arthur* testează capacitatea intelectuală a subalternilor săi. Anul acesta vrăjitorul *Merlin* va fi așezat într-un labirint a cărui hartă se cunoaște.

Labirintul conține o mulțime de obiecte pe care *Merlin* trebuie să le colecteze și să le scoată din labirint.

Merlin este pus în dificultate deoarece vrăjile sale nu au efect nici asupra zidurilor și nici asupra obiectelor și trebuie să părăsească labirintul cât mai repede posibili.

Chiar dacă labirintul conține multe obiecte, acestea sunt mici și ușoare și transportul lor nu constituie o dificultate pentru *Merlin*.

Labirintul poate fi privit ca un caroiaj de dimensiune $m \times n$ împărțit în $m \cdot n$ celule.

În fiecare minut, *Merlin* se poate deplasa din celula în care se află într-o alta care poate fi liberă sau poate conține unul dintre obiecte, vecină pe orizontală sau pe verticală. *Merlin* poate colecta din mers obiectele, deci timpul necesar colectării obiectelor este 0.

Fiecare celulă a labirintului poate conține cel mult un obiect. Celula în care se află inițial *Merlin*, este goală.

Merlin va putea părăsi labirintul dacă ajunge într-o celulă care se află pe marginea acestuia și a colectat toate obiectele.

În cazul în care *Merlin* nu reușește să iasă din labirint în cel mai scurt timp posibil, regele *Arthur* îl va concedia.

Pentru a se asigura că nu va da greș, *Merlin* vă roagă să realizați un program care, pe baza labirintului și a unei celule inițiale, să determine timpul minim în care el poate colecta obiectele și poate ieși din labirint.

Date de intrare

Prima linie a fișierului de intrare **COLLECT.IN** conține valorile m și n care reprezintă dimensiunile labirintului.

Fiecare dintre următoarele m linii conține un șir format din n caractere care pot avea valorile *W*, *O*, *E* și *M*.

Litera *W* indică faptul că în labirint, în celula respectivă este zid. Litera *O* indică faptul că în labirint, în celula respectivă se află un obiect. Litera *E* indică faptul că în labirint celula respectivă este goală. Litera *M* indică faptul că în labirint celula respectivă reprezintă poziția inițială a lui *Merlin*.

În fișierul de intrare va exista un singur caracter cu valoarea *M*.

Date de ieșire

Fișierul de ieșire **COLLECT.OUT** trebuie să conțină o singură linie pe care se va afla numărul minim de minute necesare pentru ca *Merlin* să adune toate obiectele și să părăsească labirintul.

Restricții și precizări

- $2 \le m, n \le 50$;
- *Merlin* nu va putea distruge zidurile;
- labirintul conține cel mult cinci obiecte;
- se garantează faptul că *Merlin* va putea ieși din labirint.

Exemplu

`COLLECT.IN`

```
5 5
EMEEE
WWWOW
EEOEW
EOEEE
WWWWW
```

`COLLECT.OUT`

```
9
```

Timp maxim de execuție/test: 1 secundă

P030432: Lego

Se consideră o tablă triunghiulară ale cărei laturi au lungimea n (număr întreg), împărțită în triunghiuri ca în figura alăturată și se consideră m triunghiuri care au dimensiunile laturilor egale cu 1. Pe fiecare latură a unui triunghi mic este înscris un număr.

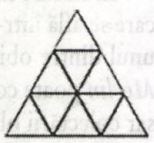


Figura 1: $n = 3$

Se cere să se așeze triunghiuri mici pe tablă astfel încât:

- tabla să fie acoperită complet;
- două laturi alăturate care sigur aparțin la două triunghiuri diferite să aibă înscrise același număr;
- suma tuturor numerelor înscrise pe laturile triunghiurilor așezate pe tablă să fie maximă.

Date de intrare

Prima linie a fișierului de intrare `LEGO.IN` conține valorile n și m, separate printr-un singur spațiu, care reprezintă lungimile celor trei laturi ale tablei, respectiv numărul de triunghiuri care au lungimile laturilor egale cu 1.

Fiecare dintre următoarele m linii conține câte trei numere, separate între ele prin spațiu, care reprezintă numerele înscrise pe laturile unui triunghi.

Date de ieșire

Fișierul de ieșire `LEGO.OUT` trebuie să conțină o singură linie pe care se va afla suma tuturor numerelor înscrise pe laturile triunghiurilor așezate pe tablă, astfel încât să fie respectate condițiile din enunț.

Restricții

- $2 \le n \le 5$;
- $3 \le m \le 30$;
- numerele înscrise pe laturile triunghiurilor sunt numere întregi cuprinse între 1 și 100;
- triunghiurile pot fi rotite în orice direcție și oglindite;
- se garantează existența unei soluții.

Exemplu

`LEGO.IN`

```
2 6
3 7 8
1 2 3
1 2 7
5 2 1
2 4 4
4 1 2
```

`LEGO.OUT`

```
46
```

Timp maxim de execuție/test: 1 secundă

P030433: Cele două (tipuri de) turnuri

De foarte mult timp, cele două specii de *elfi*, *Elfii Pădurii* și *Elfii Nopții*, trăiesc în pace pe același teritoriu.

Datorită amenințărilor din partea *orcilor*, *elfii* și-au construit turnuri pentru a-și apăra teritoriul. Bineînțeles că cele două specii de *elfi* și-au construit turnurile în stil propriu. (Adică, turnurile construite de *Elfii Pădurii* sunt diferite de cele ale *Elfilor Nopții*.)

Atunci când *orcii* nu mai reprezentau o amenințare pentru *elfi*, cele două specii au hotărât de comun acord că ar fi bine să stabilească, în funcție de turnurile pe care le-au construit, teritoriul fiecărei specii.

Ei au decis că teritoriul fiecărei specii să fie determinat pe baza turnurilor, astfel încât oricare ar fi două turnuri aparținând aceleiași specii, drumul cel mai scurt dintre ele să aparțină teritoriului și teritoriul trebuie să fie cât mai mic posibil pentru a fi mai ușor de apărat în cazul apariției unui pericol. Teritoriul unei specii trebuie să conțină toate turnurile aparținând speciei respective.

Datorită faptului că acest lucru depășește inteligența *elfilor*, ei apelează la voi pentru a-i ajuta.

Sarcina voastră este de a determina numărul de turnuri ale *Elfilor Pădurii* care se află pe teritoriul *Elfilor Nopții* și numărul de turnuri ale *Elfilor Nopții* care se află pe teritoriul *Elfilor Pădurii*.

Date de intrare

Prima linie a fișierului de intrare `TOWERS.IN` conține valorile n și m, separate printr-un singur spațiu, care reprezintă numărul de turnuri ale *Elfilor Pădurii*, respectiv numărul de turnuri ale *Elfilor Nopții*.

Fiecare dintre următoarele n linii conține câte două numere, separate între ele printr-un singur spațiu, care reprezintă coordonatele la care se află un turn al *Elfilor Pădurii*.

Fiecare dintre următoarele m linii conține câte două numere, separate între ele printr-un singur spațiu, care reprezintă coordonatele la care se află un turn al *Elfilor Nopții*.

Date de ieșire

Fișierul de ieșire `TOWERS.OUT` trebuie să conțină o singură linie pe care se vor afla două numere, separate între ele printr-un singur spațiu, care reprezintă numărul de turnuri ale *Elfilor Pădurii* care se află pe teritoriul *Elfilor Nopții*, respectiv numărul de turnuri ale *Elfilor Nopții* care se află pe teritoriul *Elfilor Pădurii*.

Restricții și precizări

- turnurile de pe marginea unui teritoriu aparțin acestuia;
- nu există două turnuri la aceleași coordonate;
- numărul total al turnurilor ($m + n$) este un întreg cuprins între 2 și 1000;
- coordonatele turnurilor sunt numere întregi cuprinse între 0 și 1000.

Exemplu

TOWERS.IN

```
4 4
0 0
4 4
0 4
4 0
3 1
5 3
3 3
5 1
```

TOWERS.OUT

```
0 2
```

Timp maxim de execuție/test: 1 secundă

P030434: 008 - James Bomb

James Bomb are o nouă misiune de îndeplinit. El se află într-o clădire cu m etaje. Fiecare etaj poate fi privit ca un caroiaj având $n \times p$ celule.

Într-un anumit loc din clădire se află o bombă care explodează după t minute. *Agentul* 008 poate să dezamorseze bomba în w minute, dar mai întâi trebuie să ajungă la ea.

În fiecare minut *James Bomb* se poate deplasa cu o celulă pe orizontală (stânga/dreapta sau înainte/înapoi) și verticală (sus/jos) într-o celulă goală.

Sarcina voastră este ca, pe baza hărții clădirii, a poziției lui *James Bomb* și a poziției bombei, să stabiliți dacă *James Bomb* are timp sau nu să dezamorseze bomba.

În cazul în care *James Bomb* are timp să dezamorseze bomba, determinați timpul minim necesar în care el ajunge la bombă. În caz contrar, determinați timpul minim necesar în care *James Bomb* poate părăsi clădirea.

Date de intrare

Fișierul de intrare **JBOMB.IN** conține pe prima linie trei numere m, n și p, separate între ele prin spațiu, care reprezintă dimensiunile clădirii. A doua linie conține două numere t și w, separate printr-un singur spațiu, care reprezintă timpul în care bomba explodează, respectiv timpul în care *James Bomb* poate să dezamorseze bomba.

Cea de-a treia linie conține trei numere, separate între ele prin spații, care reprezintă coordonatele bombei.

Cea de-a patra linie conține trei numere, separate între ele prin spații, care reprezintă coordonatele lui *James Bomb*.

În continuare urmează harta celor m etaje. Fiecare etaj este descris de n linii.

Pe fiecare dintre liniile corespunzătoare descrierii unui etaj se află p numere care nu sunt separate prin spații și care pot avea valorile 0 sau 1. Valoarea 0 indică faptul că celula respectivă este liberă, iar valoarea 1 indică faptul că celula respectivă este ocupată (conține zid).

Date de ieșire

Fișierul de ieșire **JBOMB.OUT** trebuie să conțină o singură linie pe care se vor afla două numere separate între ele printr-un singur spațiu.

Primul număr va avea valoarea 1 dacă *James Bomb* are timp să dezamorseze bomba, caz în care cel de-al doilea număr reprezintă timpul minim în care *Agentul* 008 poate ajunge la bombă și valoarea 0 dacă *James Bomb* nu are timp să dezamorseze bomba, caz în care cel de-al doilea număr reprezintă timpul minim în care *Agentul* 008 poate părăsi clădirea.

Restricții și precizări

- $3 \le m, n, p \le 20$;
- $1 \le w, t \le 100$;
- poziția inițială a lui *James Bomb* și poziția bombei sunt distincte;
- *James Bomb* poate părăsi clădirea dacă se află într-o celulă liberă de pe marginea clădirii și nu are timp să dezamorseze bomba;
- se garantează existența unei soluții.

Exemplu

JBOMB.IN

```
3 3 3
5 2
3 3 2
1 2 2
011
001
111
000
100
001
111
111
101
```

JBOMB.OUT

```
1 3
```

Timp maxim de execuție/test: 1 secundă

P030435: Raze LASER

În anul 2457, în provincia *Etram* de pe planeta *Marte* au avut mai multe explozii nucleare.

Locuitorii planetei dețin o tehnologie cu ajutorul căreia pot împiedica răspândirea particulelor radioactive care sunt capabile să distrugă toate formele de viață de pe planetă.

Tehnologia lor se bazează pe două unde de tip *LASER* care, transmise din același punct pe direcții diferite distrug toate particulele radioactive care se află într ele sau le intersectează. Locuitorii planetei au ales un loc la marginea provinciei unde au instalat un dispozitiv bazat pe tehnologia prezentată anterior.

Datorită faptului că locuitorii planetei doresc ca cele două raze să fie cât mai apropiate posibil, astfel încât să nu existe pericolul răspândirii particulelor radioactive pe toată planeta, vă roagă să determinați unghiul minim pe care undele *LASER* îl pot forma.

Date de intrare

Fișierul de intrare **LASER.IN** conține numărul n al punctelor în care au avut loc explozii nucleare.

Pe fiecare dintre următoarele n linii se află câte două numere, separate printr-un singur spațiu, care reprezintă coordonatele punctelor în care au avut loc explozii nucleare relative la locul în care a fost montat dispozitivul.

Date de ieșire

Fișierul de ieșire **LASER.OUT** trebuie să conțină o singură linie pe care se va afla un singur număr care reprezintă valoarea unghiului minim care poate fi format de cele două raze laser ale dispozitivului. Unghiul va fi scris în grade cu două zecimale exacte.

Restricții și precizări

- $1 \le n \le 100000$;
- coordonatele punctelor sunt numere întregi cuprinse între 0 și 1000;
- există puncte în care au avut loc mai multe explozii;
- dispozitivul este montat în punctul de coordonate (0, 0).

Exemplu

LASER.IN

```
4
2 1
1 1
2 4
4 3
```

LASER.OUT

```
36.87
```

Timp maxim de execuție/test: 1 secundă

P030436: Sertare

Mickey și *Minnie* s-au plictisit să stea toată ziua în casă și datorită faptului că locuința lor era mare și conținea foarte multe sertare s-au gândit la următorul joc.

Se aleg $2 \cdot n$ sertare care se numerotează de la 1 la $2 \cdot n$, și n bile albe și n bile negre. *Minnie* pune câte o bilă în fiecare dintre cele $2 \cdot n$ sertare.

Mickey începe să adune bilele din sertare în ordinea în care au fost numerotate, și dacă, la un moment dat, are mai multe bile albe decât negre, atunci el câștigă runda. La fiecare rundă, bilele trebuie așezate în altă ordine.

Minnie vă roagă să determinați, pe baza numărului n, numărul total de runde pe care le poate câștiga.

Date de intrare

Fișierul de intrare **DRAWERS.IN** conține numărul n.

Date de ieșire

Fișierul de ieșire **DRAWERS.OUT** trebuie să conțină o singură linie pe care se va afla un singur număr care reprezintă numărul total de posibilități distincte de a așeza bilele în cele $2 \cdot n$ sertare astfel încât să fie respectată condiția de așezare a bilelor.

Restricție

- $1 \le n \le 500$.

Exemple

DRAWERS.IN

```
3
```

DRAWERS.OUT

```
5
```

DRAWERS.IN

```
2
```

DRAWERS.OUT

```
2
```

Timp maxim de execuție/test: 1 secundă

P030437: Rețea de calculatoare

O companie deține o rețea de calculatoare. Oricare două calculatoare pot comunica între ele în mod direct sau indirect.

Datorită faptului că rețeaua este veche și nu a fost întreținută corespunzător, conducerea dorește să știe cât de mare este riscul ca două calculatoare din rețea să nu mai poată comunica.

Compania v-a angajat să realizați un program care, pe baza legăturilor dintre calculatoare, să determine numărul minim de legături care ar trebui distruse astfel încât să existe două calculatoare care nu mai pot comunica.

Date de intrare

Fișierul de intrare **NETWORK.IN** conține pe prima linie două numere n și m, separate între ele printr-un singur spațiu, care reprezintă numărul de calculatoare din rețea, respectiv numărul de legături directe dintre ele.

Fiecare dintre următoarele m linii conține câte două numere, separate printr-un singur spațiu, care reprezintă numerele de ordine a două calculatoare între care există legătură directă.

Calculatoarele din rețea sunt numerotate de la 1 la n și între două calculatoare există cel mult o singură conexiune.

Date de ieșire

Fișierul de ieșire **NETWORK.OUT** trebuie să conțină o singură linie pe care se va afla un singur număr care reprezintă numărul minim de legături care ar trebui distruse astfel încât să existe două calculatoare care nu mai pot comunica în mod direct sau indirect.

Restricție

- $1 \le n \le 100$ și $1 \le m \le 4950$.

Exemplu

```
NETWORK.IN
4 4
1 2
1 4
2 3
3 4

NETWORK.OUT
2
```

Timp maxim de execuție/test: 1 secundă

P030438: Extratereștri

Două grupuri de extratereștri comunică între ele prin intermediul unor mesaje. Cercetătorii de la *Institutul de Cercetări Spațiale* au reușit să intercepteze mesajele și să le decodifice sub forma unor șiruri de caractere formate din litere mici ale alfabetului englezesc.

Cercetătorii au descoperit că fiecare grup transmite două mesaje unul după celălalt și că *mesajul original* este dat de cel mai lung subșir comun al celor două mesaje.

Datorită faptului că mesajele transmise sunt foarte lungi, cercetătorii vă roagă ca, pe baza a două mesaje să determinați *mesajul original*.

Date de intrare

Fișierul de intrare **ALIENS.IN** conține pe prima linie primul mesaj, iar pe cea de-a doua linie cel de-al doilea mesaj.

Date de ieșire

Fișierul de ieșire **ALIENS.OUT** trebuie să conțină pe prima linie numărul de caractere ale mesajului original, iar pe cea de-a doua linie mesajul original.

Restricții și precizări

- lungimile celor două mesaje din fișierul de intrare este un număr întreg cuprins între 1 și 2000;
- în cazul existenței mai multor soluții se va afișa doar una.

Exemplu

ALIENS.IN	ALIENS.OUT
abracadabra	7
arbadacarba	abaaaba

Timp maxim de execuție/test: 2 secunde

P030439: Matrice magică

Se consideră o matrice A de dimensiune $n \times m$ și un număr natural p. Elementele matricei A sunt numere întregi. Matricea A are proprietatea că toate sumele elementelor care se află pe o linie sau o coloană a matricei este divizibilă cu p.

O matrice B este *asemănătoare* cu matricea A dacă are următoarele proprietăți:

- matricea B are aceeași dimensiune cu matricea A;
- $|a_{ij} - b_{ij}| < p$, $\forall 1 \le i \le n$ și $\forall 1 \le j \le m$;
- suma elementelor de pe linia i, $1 \le i \le n$, a matricei B este egală cu suma elementelor de pe linia i a matricei A;
- suma elementelor de pe coloana j, $1 \le j \le m$, a matricei B este egală cu suma elementelor de pe linia j a matricei A.

Spunem că matricea B este *magică* dacă b_{ij}, $\forall 1 \le i \le n$ și $\forall 1 \le j \le m$, este divizibil cu p și $b_{ij} \ge 0$.

Sarcina voastră este să determinați o matrice B *magică*, dacă există, *asemănătoare* cu matricea A.

Date de intrare

Fișierul de intrare **MAGIC.IN** conține pe prima linie două numere n și m, separate între ele printr-un singur spațiu, care reprezintă dimensiunea matricei A. Cea de-a doua linie conține numărul p.

Fiecare a i-a linie dintre următoarele n linii conține câte m numere, separate între ele prin spațiu, care reprezintă elementele de pe a i-a linie din matricea A.

Date de ieșire

Fișierul de ieșire **MAGIC.OUT** trebuie să conțină n linii. Fiecare a i-a dintre cele n linii trebuie să conțină m numere care reprezintă elementele de pe a i-a linie a matricei B determinate pe baza condițiilor impuse de enunțul problemei.

Dacă nu există o matrice B care să respecte condițiile din enunț, atunci în fișierul de ieșire se va scrie pe o singură linie valoarea -1.

Restricții

- $1 \le n, m \le 30$;
- $1 \le p \le 100$;
- $1 \le a_{ij} \le 100$, $\forall 1 \le i \le n$ și $\forall 1 \le j \le m$.

Exemplu

```
MAGIC.IN
3 3
3
4 5 6
7 8 9
10 11 12

MAGIC.OUT
3 6 6
6 9 9
12 9 12
```

Timp maxim de execuție/test: 5 secunde

Sfaturi pentru viitori olimpici (2)

OPTIMIZAREA programelor

Mihai Stroe

Dacă în numărul anterior am tratat problema depanării și testării programelor, în acest articol vom trata problema optimizării acestora, necesară în situația în care programele depășesc cu puțin limita de timp precizată în enunțurile problemelor care trebuie rezolvate.

Acest articol tratează optimizarea programelor. Pentru a obține efectul maxim din acest articol, este recomandată parcurgerea articolului din numărul precedent, referitor la testarea și depanarea programelor.

Cele două articole sunt bazate pe experiența personală a autorului, pe observațiile mai multor elevi și studenți participanți la concursurile de informatică și pe materialele de pregătire puse la dispoziție de organizatorii *USA Computing Olympiad* (**www.usaco.org**).

Aceste articole se adresează în special participanților la concursurile de informatică, ținând cont de specificul acestora (timp de lucru destul de mic, probleme cu enunț clar etc.). O mare parte din observațiile conținute pot fi aplicate și în alte contexte, cum al fi cel al dezvoltării de *software*; pe de altă parte, date fiind situațiile mai complexe din aceste contexte, există multe tehnici pe care nu le-am descris aici. Există posibilitatea de a aborda mai pe larg aceste subiecte într-un articol viitor.

Toate articolele din numerele anterioare ale revistei, care sunt menționate în această expunere, se găsesc în format *pdf* pe *site*-ul revistei, **www.ginfo.ro**.

Introducere

A optimiza un program înseamnă a-l modifica pentru a obține un program mai rapid.

Prima observație este ca un program mai rapid, dar care dă rezultate incorecte, este în cele mai multe cazuri inutil. De aceea, se recomandă ca, la rezolvarea unei probleme, să se scrie programe clare și ușor de depanat.

Aceste aspecte sunt, în prima fază, mai importante decât viteza de execuție. Este preferabil să avem un program care funcționează corect pe care să îl optimizăm, față de un program greu de depanat și care produce rezultate eronate (chiar dacă le produce repede).

Pentru a avea un program rapid este esențial să avem un algoritm rapid. Astfel, un program care implementează un algoritm cu ordinul de complexitate $O(N^3)$ va rula, de obicei, mult mai încet decât unul cu ordinul de complexitate $O(N^2)$ pentru valori mari ale lui N, indiferent cât de mult îl vom optimiza. De aceea, găsirea unui algoritm bun de rezolvare este mai importantă decât optimizarea.

Nici acest pas nu trebuie dus la extrem. Dacă un algoritm mai lent, dar ușor de implementat, se va încadra în timp, acesta este preferabil algoritmului optim; în acest caz dispare necesitatea optimizării.

În continuare vom presupune că se optimizează un program de dimensiune medie sau mare, care rezolvă o problema dată în concurs. Optimizarea unui program de 10-15 linii poate fi tratată mult mai simplu.

Pe parcursul etapei de optimizare se recomandă păstrarea tuturor versiunilor corecte ale programului, prin salvarea lor cu alte nume. Se elimină astfel necesitatea rescrierii unor mari părți de cod pentru a reveni la o variantă anterioară, daca se dovedește că optimizările curente nu funcționează. Se recomandă ca numele unei versiuni să conțină și numărul acesteia, pentru a avea o imagine mai clară a etapelor din evoluția programului.

În același scop se poate folosi comentarea secțiunilor de cod care vor fi optimizate. Recomand această variantă pentru înlocuirea unor secțiuni reduse ca dimensiuni. Un program cu multe comentarii mari este mai greu de modi-

ficat, deoarece trebuie să folosim *PageUp - PageDown* de mai multe ori pentru a naviga între secțiuni. În plus, dacă totuși alegeți varianta comentării blocurilor mari, recomand inserarea în comentarii a unor informații care să permită revenirea rapidă la o versiune funcțională anterioară.

Programul se va optimiza până în momentul în care apreciem că se va încadra în timp pentru testele cele mai dificile pe care le putem construi. Un program care merge instantaneu nu va primi mai multe puncte decât un program care rulează foarte aproape de limita de timp, dar nu o depășește niciodată; optimizarea celui de-al doilea program este inutilă, iar timpul care ar fi alocat acesteia poate fi folosit în alte scopuri.

Pentru a demonstra importanța optimizării, amintesc o realizare a unei echipe a *Universității "Politehnica"* din *București*, din care am făcut parte, la etapa regională a concursului *ACM* din anul 2000.

Echipa a rezolvat o problemă dificilăcu un algoritm de complexitate $O(N^2)$, pentru care soluția oficială (pe care nu am reușit să o găsim) avea ordinul de complexitate $O(N \cdot \log N)$, folosind tehnicile de optimizare prezentate în acest articol.

Rezolvarea inițială, neoptimizată, nu se încadra în timp, motiv pentru care problema a fost abandonată... până în ultimele 7 minute ale probei, când nu mai era nimic altceva de făcut. Totuși, problema a fost rezolvată; au fost folosite structuri de date mai eficiente, o sortare ineficientă a fost înlocuită cu *QuickSort*, după sortare au fost eliminate duplicatele, s-au făcut câteva optimizări de cod etc.

Se pare că nici o altă echipă nu a rezolvat problema respectivă.

Ca fapt divers, am fi obținut locul 1 și fără a rezolva această problemă, dar în alte situații optimizarea unui program poate fi decisivă pentru obținerea unei performanțe. Există multe exemple în acest sens.

Optimizări în cadrul algoritmului

Există două reguli importante pentru optimizarea în cadrul unui algoritm:

- *Don't Do Anything Stupid! – Nu face nimic aiurea!*
- *Don't Do Anything Twice! – Nu face un lucru de două ori!*

Pentru a respecta aceste reguli este necesar să identificăm ce anume se face de mai multe ori și ce anume este inutil. Vom da exemple pentru ambele situații.

Pentru a nu calcula același lucru de două ori, folosim tehnici de preprocesare. Acestea sunt prezentate pe larg într-un articol din numărul din martie 2003 al *GInfo* și nu vor fi reluate în această expunere.

De asemenea, în multe cazuri este recomandată eliminarea valorilor duplicate dintr-o anumită structură, sau compactarea anumitor secvențe.

Felul în care se va evita lucrul inutil într-un program se poate exemplifica în multe feluri. Un exemplu foarte clar constă în generarea numerelor prime; pentru a determina dacă un număr N este prim îl vom împărți nu la toate numerele mai mici ca N, ci la toate numerele mai mici sau egale cu $\sqrt{N}$. Dacă generăm și memorăm toate numerele prime începând de la 2, atunci la testarea numărului N îl vom împărți la toate numerele prime mai mici sau egale cu $\sqrt{N}$. De asemenea, nu vom testa numerele pare mai mari decât 2.

Faptul că, pentru o anumită problemă, aveți un algoritm de complexitate $O(N^2)$ nu înseamnă că trebuie ca toți pașii să fie $O(N^2)$. De exemplu, transformarea unei sortări din *BubbleSort* în *QuickSort* poate duce la un câștig important de viteză, chiar dacă un pas ulterior al algoritmului rămâne $O(N^2)$. Generalizând, dacă putem optimiza anumiți pași ai rezolvării, este bine să o facem, chiar dacă ordinul final de complexitate rămâne neschimbat.

În multe probleme, spațiul soluțiilor conține soluții simetrice. De exemplu, în cazul problemei găsirii tuturor variantelor de așezare a N dame pe o tablă de șah astfel încât să nu se atace reciproc, se pot obține foarte ușor soluții simetrice cu o anumită soluție. Încercați să folosiți simetriile pentru a reduce timpul de execuție, parcurgând o parte mai mică din spațiul soluțiilor și afișând soluțiile simetrice. Aveți grijă la soluțiile simetrice cu ele însele, deoarece acestea nu trebuie afișate de mai multe ori.

În unele probleme, este util să descompunem spațiul analizat. De exemplu, multe din problemele *NP* pe grafuri se pot rezolva independent pe componentele conexe ale grafului; luarea în considerare a acestui fapt duce la o creștere substanțială a vitezei.

Există probleme care se pot rezolva mai eficient dacă datele sunt procesate în altă ordine față de cea din fișierul de intrare. Gândiți-vă la modul în care este afectată rezolvarea dacă datele de intrare sunt sortate, parcurse în ordine inversă, sau parcurse de mai multe ori în ordine aleatoare.

Pentru metode eficiente de abordare a problemelor *NP*, studiați articolul din numărul din octombrie 2003 al *GInfo*. Sunt prezentate idei ca: simplificarea problemei, eliminarea explorării situațiilor neinteresante, explorarea *"Depth First Search with Iterative Deepening"* etc.

Există probleme pentru care anumiți pași se pot rezolva cu complexitate logaritmică, în loc de complexitate liniară. Pentru mai multe detalii, studiați articolul pe această temă apărut în *GInfo* în noiembrie 2003.

În problemele în care apar calcule cu numere mari (cu câteva zeci/sute de cifre), de multe ori se folosește un vector de cifre pentru memorarea numerelor mari. Elementele acestui vector sunt numere între 0 și 9 inclusiv.

O variantă de optimizare a operațiilor cu numere mari constă în a lucra în baza 10^K. Astfel, elementele vectorului

care reprezintă un număr mare vor fi întregi între 0 și 10^K-1 inclusiv. Evident, un număr cu C cifre va corespunde unui vector cu C/K elemente, deci se vor efectua mai puține operații; astfel, adunarea a două numere mari constă în procesarea a C/K poziții în loc de C poziții etc.

Se recomandă mare atenție la afișarea numerelor mari. Astfel, de exemplu, numărul 909 este reprezentat în baza 100 prin două elemente 9; trebuie avut grijă să nu afișăm 99.

Unii autori recomandă folosirea unor baze de numerație care să fie puteri ale lui 2, cum ar fi 256. Această abordare oferă o eficiență ridicată, dar programele obținute pot deveni greu de depanat. Personal, consider că varianta cu baza 10^K este satisfăcătoare pentru valori bine alese ale lui K.

O ultimă optimizare menționată aici constă în preferarea algoritmilor care lucrează cu numere întregi, față de cei pe numere reale. Motivul este simplu: operațiile cu numere întregi sunt mai rapide.

De exemplu, dacă datele de intrare sunt numere reale cu cel mult trei cifre înainte de virgulă și cel mult două cifre după, ele pot fi transformate în întregi de cinci cifre prin înmulțirea cu 100, după care toate operațiile sunt mult mai rapide.

Există și exemple de probleme în care transformarea nu este la fel de evidentă.

Optimizări de cod

Aceste optimizări se referă la modificări minore aduse codului sursă, cu rezultate importante.

Prima categorie de astfel de optimizări ține de preprocesare, tehnica amintită anterior. Astfel, dacă valoarea unei expresii este folosită de mai multe ori, ea poate fi memorată într-o variabilă, pentru a nu fi recalculată de fiecare dată. Mai multe detalii se găsesc în articolul dedicat acestui subiect.

De exemplu, în C/C++, în locul secvenței:

```
for(i=2;i <= sqrt(n);i++)
{
  //prelucrări
}
```

este indicat să se folosească secvența:

```
radical <= sqrt(n);
for(i=2;i <= radical;i++)
{
  //prelucrări
}
```

O categorie de optimizări de cod care nu ține de codul efectiv constă în alegerea opțiunilor de compilare. Astfel, pe parcursul testării și depanării programului se folosesc opțiuni recomandate în acest sens, în timp ce în etapa optimizării testarea se realizează folosind opțiunile "de viteză". De exemplu, în *Pascal*, verificările de tipul *Range Checking* (*Verificarea depășirii domeniului* – `$R`) și *Stack Checking* (*Verificarea depășirii stivei* – `$S`) sunt setate pe minus.

Apelurile de funcții și proceduri introduc un timp suplimentar pentru alocarea pe stivă a spațiului pentru parametri, variabile locale etc. Dacă o funcție este apelată de puține ori, acest timp este neglijabil, dar pentru o funcție scurtă care se apelează de un milion de ori, timpul suplimentar începe să conteze. În aceste cazuri este recomandată scrierea ***inline*** a codului funcției. Dimensiunea programului poate crește, deoarece aceeași funcție poate fi apelată în mai multe locuri, cu parametri diferiți. De aceea, în multe cazuri, în etapa de implementare și depanare se va folosi funcția; trecerea la varianta *inline* se va face doar dacă programul nu este suficient de rapid.

Evident, dacă, în varianta neoptimizată a programului, o funcție este apelată de mai multe ori în același context (aceiași parametri etc.) se va folosi preprocesarea și se vor reține valorile deja calculate.

Dacă nu se dorește scrierea *inline* a unei funcții, o metodă de optimizare constă în reducerea numărului de parametri și variabile locale.

Folosirea tipurilor de date adecvate influențează viteza algoritmului. De exemplu, dacă avem un vector de întregi între 0 și 25 și îl inițializăm de multe ori cu anumite valori, îl vom declara vector cu elemente de tipul `byte` (în *Pascal*) și nu `longint`. Astfel, operațiile care implică întreaga structură se vor realiza mai rapid, deoarece zona de memorie accesată va fi mai mică.

Există și cazuri în care folosirea unui tip de date cu un domeniu mai mare poate duce la creșterea vitezei. Se recomandă testarea diferitelor variante.

Limbajul C/C++ oferă multe optimizări de cod care nu există în *Pascal*. Vom examina câteva dintre ele.

În aceste limbaje există funcții de genul `memcpy()`, care copiază eficient un bloc de memorie. Este recomandată folosirea acestor funcții, față de varianta în care copiem, pe rând, fiecare element. În *Pascal* există procedura similară `fillchar` pentru inițializarea unei zone de memorie, se permite copierea integrală a unui vector peste altul, dar nu există funcții și proceduri eficiente pentru copierea unei anumite zone.

De fapt, în general, dacă limbajul de programare pune la dispoziție o anumită funcție pentru a realiza o anumită operație, folosirea funcției respective este preferabilă scrierii unei funcții proprii.

În unele probleme de *programare dinamică* pe matrice putem memora doar ultimele două linii ale matricei, ele-

mentele din linia *i* calculându-se pe baza celor din linia *i*-1. Fie aceste linii reprezentate în doi vectori, *A* și *B*. De obicei, valorile din *B* sunt calculate pe baza celor din *A*, după care *A* este suprascris cu *B* și se trece la pasul următor.

Acest pas se poate face mult mai ușor dacă pentru *A* și *B* folosim *point*-eri; astfel, copierea constă într-o interschimbare de *point*-eri, deoarece valorile care se aflau în *B* vor fi oricum suprascrise. Pe de altă parte, în *Pascal* folosirea *point*-erilor pe parcursul rezolvării poate duce la creșterea timpului de execuție.

Se recomandă testarea ambelor variante și alegerea celei mai eficiente. De fapt, există o variantă care elimină ambele neajunsuri; lăsăm găsirea ei ca exercițiu pentru cititor.

Înmulțirile și împărțirile cu 2 (sau 2 la o anumită putere) se pot realiza prin deplasări pe biți la stânga, respectiv la dreapta. Pentru a ușura depanarea programului, se recomandă scrierea acestor operații în mod normal pentru etapa de testare și depanare și transformarea lor în deplasări numai dacă sunt necesare optimizări. Programatorii cu experiență pot ignora cu succes această recomandare.

Dacă o problemă se poate rezolva folosind ca structuri de date vectori, într-o primă variantă, și matrice, într-o a doua variantă, prima variantă este de preferat (presupunând că restul condițiilor, cum ar fi ordinul de complexitate, sunt identice). Motivul este următorul: pentru accesarea elementelor unei matrice se fac mai multe calcule cu indicii, decât pentru accesarea elementelor unui vector.

De asemenea, folosirea structurilor de tip înregistrare (**record** în *Pascal*, **struct** în *C/C++*) duce la încetinirea programului.

Astfel, din următoarele 3 variante de definire ale unui vector de puncte în plan (fiecare punct este definit prin două coordonate întregi, *x* și *y*):

```
• type punct = record
                 x,y: integer;
               end;
  var puncte: array[1..1000] of punct;

• var puncte: array[1..1000,1..2] of integer;

• var x,y: array[1..1000] of integer;
```

dacă eficiența contează, se va alege, în cele mai multe cazuri, cea de-a treia variantă.

În *C/C++*, unele variabile pot fi prefixate cu modificatorul **register**. Acesta îi spune compilatorului că variabila respectivă va fi folosită frecvent, deci accesul la ea ar trebui optimizat. Concret, de obicei variabila respectivă va fi păstrată într-un registru.

Opțiunea funcționează pentru variabile de tip `int` și `char`. Nu pot fi păstrate foarte multe variabile în regiștri în același timp; dacă sunt prea multe variabile declarate, de tip **register**, compilatorul va ignora o parte din declarații.

Se recomandă folosirea acestui modificator în bucle, de exemplu, pentru variabila contor dintr-un ciclu **for**.

Tot în *C/C++*, folosirea aritmeticii *pointer*-ilor poate fi mai indicată decât folosirea indicilor într-un vector. Recomand studierea operațiilor cu *point*-eri pentru mai multe informații.

Ultima categorie de optimizări de cod amintită se referă la scrierea unor secvențe de cod în limbaj de asamblare, dacă acest lucru este permis în concurs. Din nefericire, autorul acestui articol nu este expert în acest domeniu, deci nu vă poate oferi mai multe detalii...

Concluzii

Există multe moduri de a optimiza un program. Unele din ele se referă la îmbunătățiri majore ale algoritmului, altele la îmbunătățiri minore, iar altele se referă la modificări de câteva caractere sau linii ale unor secvențe de cod.

Cunoașterea și punerea în practică a tehnicilor de optimizare prezentate în acest articol (și, eventual, a altora) pot aduce puncte prețioase, care nu s-ar fi putut obține altfel (deoarece, de exemplu, algoritmul optim de rezolvare este foarte greu de găsit, sau depășește experiența concurentului). Pe de altă parte, recomandăm insistent abordarea cu seriozitate a fazelor de găsire a algoritmului eficient, de implementare, de testare și de depanare. Fără aceasta, nu avem ce optimiza!

Mihai Stroe este redactor al GInfo și masterand la Universitatea Politehnica din București. El poate fi contactat prin e-mail la adresa mihai_stroe@yahoo.com.

Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

În cadrul acestui articol vom prezenta un set de funcții oferite de limbajul PHP pentru prelucrarea fișierelor XML și vom exemplifica acest set de funcții.

PHP și XML

Limbajul *PHP* oferă un set de funcții și clase care sunt utile în cazul în care se dorește prelucrarea sau accesarea fișierelor *XML*. Setul de clase este realizat în conformitate cu specificațiile *DOM Level* 1 (*Document Model Object – Model obiectual de document*) care se referă atât la fișierele de tipul *XML*, cât și la fișierele de tipul *HTML*.

După cum probabil știți, un fișier *XML* are o structură similară unui fișier *HTML* și se deosebește de acesta prin faptul că secțiunile pot avea orice nume și orice atribute și fiecare secțiune are un identificator de sfârșit al secțiunii.

Un fișier *XML*, ca de altfel și un fișier *HTML*, pot fi privite ca o structură arborescentă de date.

Elementele următoare caracterizează această structură arborescentă:

- fiecare secțiune, mai puțin secțiunea rădăcină, are un părinte;
- fiecare secțiune, mai puțin secțiunea rădăcină, are unul sau mai mulți frați (un frate al unei secțiuni este o secțiune care are același părinte cu aceasta);
- fiecare secțiune poate avea 0 sau mai mulți fii;
- fiecare secțiune are un tip și fiecare tip de secțiune are diferite atribute;
- structura unui fișier *XML* poate fi controlată cu ajutorul specificațiilor *DTD* (*Document Type Definition – Definiția tipului documentului*); aceste definiții se pot afla în fișiere externe sau chiar în cadrul fișierului *XML*.

În principiu, un document realizat în conformitate cu specificațiile *DOM* are două elemente componente, și anume restricțiile pe care trebuie să le respecte structura arborescentă și această structură. În acest moment se poate construi un nod care în cadrul specificațiilor *DOM* are tipul *DomDocument* și care să fie rădăcina întregii structuri arborescente descrisă de acest document. Acest nod are cel mult doi fii și anume un fiu este constituit de secțiunea principală descrisă în document, iar cel de-al doilea fiu este constituit din restricțiile pe care trebuie să le respecte primul fiu. Cei doi fii ai nodului rădăcină sunt de tipul *DomElement* și *DomDocumentType*.

În cadrul specificațiilor *DOM*, orice nod (secțiune) al structurii arborescente este derivat din tipul *DomNode*, deci și nodul de tipul *DomDocument* este un tip derivat din *DomNode*.

În cadrul specificațiilor *DOM* există mai multe tipuri de noduri și funcționalitatea fiecărui tip de nod este dată de o clasă în cadrul limbajului *PHP*.

În continuare vom prezenta clasele *DomNode* și *DomDocument*.

Clasa DomNode

Această clasă încapsulează proprietățile comune ale oricărui nod din structura arborescentă și, în concluzie, stă la baza tuturor tipurilor de noduri pe care arborele documentului le poate conține.

În continuare vom prezenta metodele și proprietățile acestei clase și menționăm că nici una dintre metode nu poate fi apelată static.

Metoda DomNode->node_name()

Această proprietate nu are parametri și returnează un șir de caractere care reprezintă numele unui nod din arbore. Există tipuri de noduri al căror nume nu este utilizat și a cărui modificare nu este luată în considerare sau este generată o eroare. De exemplu, pentru un nod de tipul *DomDocument*, numele este întotdeauna "*#document*".

Metoda DomNode->node_type()

Această metodă nu are parametri și returnează un număr întreg care reprezintă tipul unui nod din arbore. Pentru fiecare tip de nod este definită câte o constantă.

În tabelul 1 pot fi observate numele constantelor oferite de limbajul *PHP* pentru diferitele tipuri de noduri care apar în cadrul specificațiilor *DOM*.

Metoda DomNode->node_value()

Această metodă returnează un șir de caractere care reprezintă valoarea (informațiile) asociată nodului curent. Există tipuri de noduri care nu au valoare.

Metoda DomNode->add_namespace()

Această metodă setează pentru nodul curent domeniul de utilizare (*namespace*) și primește doi parametri de tipul șir de caractere. Primul parametru reprezintă identificatorul resursei constituite de acest nod (*URI - Uniform Resource Identifier*), iar cel de-al doilea parametru reprezintă prefixul spațiului de utilizare.

Metoda DomNode->set_namespace()

Această metodă modifică domeniul de utilizare a nodului curent și primește doi parametri de tipul șir de caractere. Parametrii au aceeași semnificație ca la metoda anterioară, iar cel de-al doilea parametru este opțional.

Metoda DomNode->prefix()

Metoda `prefix()` nu are parametri și returnează un șir de caractere care reprezintă prefixul spațiului de utilizare al nodului curent.

Metoda DomNode->parent_node()

Această metodă returnează un obiect derivat din *DomNode* care reprezintă părintele nodului curent. În structura arborescentă asociată unui document există un singur nod care nu are părinte, și anume nodul de tipul *DomDocument* care este părintele întregului arbore.

Metoda DomNode->parent()

Această metodă nu primește nici un parametru și returnează un obiect derivat din *DomNode* care reprezintă părintele nodului curent. În structura arborescentă asociată unui document există un singur tip nod care nu are părinte, și anume *DomDocument* care este părintele întregului arbore.

Metoda Dom...->owner_document()

Metoda `owner_document()` nu are parametri și returnează un obiect de tipul *DomDocument* care reprezintă părintele structurii căreia îi aparține nodul curent.

Metoda DomNode->clone_node()

Această funcție are un singur parametru de tip logic și realizează duplicare conținutului nodului curent. Dacă parametrul are valoarea **TRUE**, atunci este duplicat întregul subarbore care are ca părinte nodul curent.

În cazul în care parametrul are valoarea **FALSE** sau lipsește, se duplică numai tipul nodului și atributele.

Metoda `clone_node()` returnează un obiect de tipul celui curent.

Metoda DomNode->has_child_nodes()

Această metodă nu are parametri și returnează o valoare logică indicând faptul că nodul curent are sau nu fii.

Atributele unui nod sunt accesate folosind alte metode pe care le vom prezenta în cadrul acestui articol.

Există tipuri de noduri care sunt frunze ale structurii arborescente, deci care nu pot avea fii.

Metoda DomNode->first_child()

Metoda `first_child()` nu are parametri și returnează un obiect care reprezintă primul fiu al nodului curent, dacă există, și valoarea logică **FALSE** în caz contrar.

Metoda DomNode->last_child()

Metoda `last_child()` nu are parametri și returnează un obiect care reprezintă ultimul fiu din lista de fii a nodului curent, dacă un astfel de fiu există, și valoarea logică **FALSE** în caz contrar.

Metoda DomNode->child_nodes()

Metoda `child_nodes()` nu are parametri și returnează un tablou care conține toți fiii nodului curent.

În cazul în care nodul curent nu are fii, metoda returnează valoarea logică **FALSE**.

Constantă	Valoare	Clasă asociată
XML_ELEMENT_NODE	1	DomElement
XML_ATTRIBUTE_NODE	2	DomAttribute
XML_TEXT_NODE	3	DomText
XML_CDATA_SECTION_NODE	4	DomCData
XML_ENTITY_REF_NODE	5	DomEntityReference
XML_ENTITY_NODE	6	DomEntity
XML_PI_NODE	7	DomProcessingInstruction
XML_COMMENT_NODE	8	DomComment
XML_DOCUMENT_NODE	9	DomDocumet
XML_DOCUMENT_TYPE_NODE	10	DomDocumentType
XML_DOCUMENT_FRAG_NODE	11	DomDocumentFragment
XML_NOTATION_NODE	12	DomNotation

Tabelul 1: Constantele asociate diferitelor tipuri de noduri

Metoda DomNode->append_child()

Această metodă primește ca parametru un obiect care are un tip derivat din *DomNode* și îl adaugă la sfârșitul listei de fii.

Metoda DomNode->insert_before()

Această metodă inserează un obiect în lista de fii.

Metoda primește două obiecte ca parametru.

Primul parametru reprezintă obiectul care va fi inserat, iar al doilea reprezintă o referință spre obiectul înaintea căruia se va face inserarea.

Metoda DomNode->replace_child()

Această metodă se folosește pentru a schimba valoarea unui fiu cu un alt obiect.

Metoda are doi parametri de tip obiect. Primul parametru reprezintă o referință spre obiectul care este fiu al nodului curent, iar cel de-al doilea parametru reprezintă obiectul cu care fiul va fi înlocuit.

Metoda DomNode->remove_child()

Metoda `remove_child()` se folosește pentru a elimina un nod al fiului curent.

Metoda primește ca parametru o referință spre fiul care va fi eliminat din listă.

Metoda DomNode->has_attributes()

Această metodă nu are parametri și returnează o valoare logică indicând faptul că nodul curent are sau nu atribute.

Există tipuri de noduri care nu au atribute.

Metoda DomNode->attributes()

Metoda `child_nodes()` nu are parametri și returnează un tablou care conține obiecte de tipul *DomAttribute* care reprezintă atributele nodului curent. În cazul în care nodul curent nu are atribute, metoda returnează valoarea logică **FALSE**.

Metoda DomNode->previous_sibling()

Acestă metodă nu are parametri și returnează un obiect care reprezintă nodul care se află în lista de fii a părintelui înaintea nodului curent.

Dacă nodul curent este primul nod din lista de fii a părintelui său, atunci funcția returnează valoarea logică **FALSE**.

Metoda DomNode->next_sibling()

Acestă metodă nu are parametri și returnează un obiect care reprezintă nodul care se află în lista de fii a părintelui după nodul curent.

Dacă nodul curent este ultimul din lista de fii a părintelui său, atunci funcția returnează valoarea logică **FALSE**.

Metodele pe care le-am prezentat anterior care se referă la fiii și frații unui nod al arborelui unui document pot fi utilizate pentru a naviga în interiorul structurii arborescente.

Clasa DomDocument

Această clasă este derivată din clasa *DomNode* și conține metode necesare pentru a crea diferite tipuri de noduri care pot fi atașate ulterior structurii arborescente, pentru a procesa restricțiile pe care trebuie să le respecte arborele unui document.

Limbajul *PHP* oferă funcții pentru a crea o structură arborescentă vidă sau o structură arborescentă a unui document stocat într-un fișier pe disc sau într-un șir de caractere. În continuare vă prezentăm aceste funcții, urmând ca după aceea să prezentăm metodele clasei *DomDocument*.

Funcția domxml_new_doc()

Această funcție creează un document *XML* nou și returnează un obiect de tipul *DomDocument*. Funcția primește un singur parametru de tipul șir de caractere care reprezintă versiunea tipului de fișier *XML* și o valoare posibilă pentru acesta este `"1.0"`.

Funcția domxml_open_file()

Această funcție creează arborele unui document *XML* pe baza informațiilor stocate într-un fișier și returnează un obiect de tipul *DomDocument*.

Funcția domxml_open_mem()

Această funcție creează arborele unui document *XML* pe baza informațiilor stocate într-un șir de caractere și returnează un obiect de tipul *DomDocument*.

Metoda DomDocument->create_attribute()

Metoda `create_attribute()` creează și returnează un obiect de tipul *DomAttribute* care reprezintă un atribut care poate fi asociat unui nod din structura arborescentă.

Această metodă primește ca parametru două șiruri de caractere. Primul parametru reprezintă numele atributului care va fi creat, iar cel de-al doilea reprezintă valoarea acestuia.

Metoda DomDocument->create_cdata_section()

Această metodă creează și returnează un obiect de tipul *DomCData* care reprezintă un nod a cărui valoare este de tip șir de caractere și al cărui nume este `"#cdata-section"`.

Metoda `create_cdata_section()` are un singur parametru de tip șir de caractere care va constitui valoarea nodului care va fi creat.

Acest nod poate fi atașat structurii arborescente prin intermediul metodelor `append_child`, `insert_before` sau `replace_child` apelată pentru un nod din arbore.

În cadrul unui fișier *XML* acest tip de nod este reprezentat astfel:

```
<![CDATA[valoare]]>
```

unde `valoare` reprezintă valoarea nodului de tipul *DomCData*.

Metoda DomDocument->create_text_node()

Această metodă creează și returnează un obiect de tipul *DomText* care reprezintă un nod a cărui valoare este de tip șir de caractere și al cărui nume este `"#text"`.

Metoda `create_text_node()` are un singur parametru de tip șir de caractere care va constitui valoarea nodului care va fi creat.

Tipul de nod *DomText* este derivat de tipul *DomCData* și diferă prin acesta numai prin modul în care sunt reprezentate datele într-un fișier.

Metoda DomDocument->create_comment()

Metoda `create_comment()` creează și returnează un obiect de tipul *DomComment* care reprezintă un comentariu în structura arborescentă și primește ca parametru un șir de caractere care reprezintă textul care va fi asociat nodului care va creat.

Tipul de nod *DomComment* este derivat de tipul *DomCData* și diferă prin acesta numai prin modul în care sunt reprezentate datele într-un fișier.

În cadrul unui fișier *XML* acest tip de nod este reprezentat astfel:

```
<!-- valoare -->
```

unde `valoare` reprezintă valoarea nodului de tipul *DomComment*.

Metoda DomDocument->create_element()

Această metodă creează și returnează un obiect de tipul *DomElement*. Metoda `create_element()` are un singur parametru de tip șir de caractere care reprezintă numele nodului de tipul *DomElement* care va fi creat.

Acest tip de nod nu are valoare dar poate avea atribute și poate avea fii și într-un document *XML* este reprezentat astfel:

```
<nume atr1="val1" atr2="val2" ... atrn="valn" />
```

sau

```
<nume atr1="val1" atr2="val2" ... atrn="valn">
  reprezentare fii
</nume>
```

unde `nume` reprezintă numele nodului de tipul *DomElement*, `atr1`, `atr2`, ..., `atrn` reprezintă atributele nodului și `val1`, `val2`, ..., `valn` reprezintă valorile asociate atributelor.

Dacă un nod de tipul *DomElement* nu are fii este folosită prima sintaxă și în caz contrar cea de-a doua.

Metoda DomDocument->create_element_ns()

Această metodă creează și returnează un obiect de tipul *DomElement* căruia i se va asocia un nume, un identificator de resursă și un domeniu de utilizare.

Metoda `create_element_ns()` are trei parametri de tip șir de caractere care reprezintă identificatorul resursei constituită de acest obiect, numele nodului, respectiv prefixul domeniul de utilizare al nodului.

Metoda DomDocum...->create_entity_reference()

Metoda `create_entity_reference()` creează și returnează un obiect de tipul *DomEntityReference*. Această metodă primește ca parametru un șir de caractere care reprezintă valoarea asociată unui nod de tipul *DomEntityReference*.

Un astfel de nod nu are nume, ci doar valoare, nu poate avea atribute și nici fii, deci putem considera că acest tip de nod poate fi doar o frunză a arborelui unui document.

În cadrul unui fișier *XML* un nod de tipul *DomEntityReference* este reprezentat astfel:

```
&valoare;
```

unde `valoare` reprezintă valoarea nodului.

Metoda Dom...->create_processing_instruction()

Metoda `create_processing_instruction()` creează și returnează un obiect de tipul *DomProcessingInstruction*.

Un nod de tipul *DomProcessingInstruction* reprezintă o instrucțiune de procesare care are o *țintă* și *date*.

Această metodă are doi parametri de tip șir de caractere care reprezintă ținta instrucțiunii de procesare, respectiv datele.

În cadrul unui fișier *XML*, un nod de tipul *DomProcessingInstruction* nu poate avea fii și nici atribute.

Acest tip de nod este reprezentat într-un fișier *XML*, astfel:

```
<?țintă date?>
```

unde `țintă` reprezintă ținta instrucțiunii de procesare, iar `date` reprezintă datele instrucțiunii de procesare.

Metoda DomDocument->document_element()

Un nod de tipul *DomDocument* poate avea ca fiu un singur element de tipul *DomElement*, și pentru a ușura munca programatorului există o metodă care returnează o referință la acest element și anume `document_element()` care nu are nici un parametru.

Metoda DomDocument->doctype()

Această metodă returnează un obiect de tipul *DomDocumentType* care reprezintă tipul unui document *XML*, adică restricțiile pe care trebuie să le respecte arborele documentului dacă astfel de restricții sunt definite.

Metoda DomDocument->dump_file()

Metoda `dump_file()` nu este prevăzută în standardul *DOM*, dar ea este utilizată pentru a salva arborele unui document sub forma unui fișier *XML*.

Această metodă are trei parametri. Primul parametru este de tip șir de caractere și reprezintă numele fișierului în care se va salva structura arborescentă.

Al doilea parametru este opțional, este de tip logic și indică faptul că se va folosi sau nu compresie pentru a salva arborele documentului, iar al treilea parametru este opțional, este de tip logic și indică faptul că se va folosi sau nu formatarea documentului în momentul salvării.

Metoda returnează o valoarea logică **TRUE** dacă s-a reușit salvarea arborelui în fișierul specificat ca parametru, și valoarea logică **FALSE** în caz contrar.

Metoda DomDocument->dump_mem()

Metoda `dump_mem()` nu este prevăzută în standardul *DOM*, dar ea este utilizată pentru a salva arborele unui document sub forma unui șir de caractere.

Această metodă are doi parametri opționali, de tip logic. Acești parametri au aceeași semnificație cu ultimii doi parametri ai metodei prezentate anterior.

Metoda `dump_mem()` returnează un șir de caractere care reprezintă documentul *XML*.

Clasa DomCData și clasele derivate

Nodurile de tipul *DomCData*, *DomText* și *DomComment* nu pot avea atribute și nici fii, astfel ele sunt frunze ale arborelui unui document.

Clasele *DomText* și *DomComment*, derivate din clasa *DomCData*, nu au metode proprii, iar clasa *DomCData*, pe lângă metodele moștenite de la clasa *DomNode*, introduce metoda `length()` care returnează dimensiunea datelor conținute.

Clasa DomElement

Într-un document *XML*, în afară de restricțiile care trebuie să le respecte arborele documentului, există un singur tip de nod care poate fi nod intern. Este vorba despre *DomElement*. Tot *DomElement* este tipul de nod care poate avea atribute.

În continuare vă prezentăm metodele pe care clasa *DomElement* le introduce pe lângă cele moștenite de la clasa *DomNode*.

Metoda DomElement->has_attribute()

Această metodă are un singur parametru de tip șir de caractere și returnează valoarea logică **TRUE** dacă nodul curent are setat atributul cu numele primit ca parametru și valoarea logică **FALSE** în caz contrar.

Metoda DomElement->get_attribute_node()

Această metodă primește ca parametru un șir de caractere și returnează atributul nodului curent cu numele primit ca parametru, sub forma unui obiect de tipul *DomAttribute*. Dacă nodul curent nu are acest atribut setat, metoda returnează valoarea logică **FALSE**.

Metoda DomElement->get_attribute()

Această metodă primește ca parametru un șir de caractere și returnează un șir de caractere care reprezintă valoarea atributului nodului curent cu numele primit ca parametru. Dacă nodul curent nu are acest atribut setat, metoda returnează valoarea logică **FALSE**.

Metoda DomElement->set_attribute()

Metoda `set_attribute()` se folosește pentru a atașa nodului curent un atribut.

Metoda primește doi parametri de tipul șir de caractere. Primul parametru reprezintă numele atributului, iar al doilea reprezintă valoarea pe care acesta o va avea.

Dacă nodul curent nu are setat atributul cu numele primit ca parametru, atunci va fi creat un nod de tipul *DomAttribute* și va fi adăugat în lista de atribute a nodului curent. În caz contrar va fi schimbată valoarea pe care o are nodul de tipul *DomAttribute* corespunzător numelui primit ca parametru.

Metoda DomElement->remove_attribute()

Metoda `remove_atribute()` se folosește pentru a șterge un atribut din lista de atribute a nodului curent. Metoda primește ca parametru numele atributului care va fi șters.

În cazul în care ștergerea s-a efectuat cu succes, metoda returnează valoarea logică **TRUE**, altfel returnează valoarea logică **FALSE**.

Metoda DomElement->tagname()

Metoda `tagname()` returnează numele nodului curent și are același efect cu apelul metodei `node_name()` moștenită de la clasa de bază *DomNode*.

Metoda DomEle...->get_elements_by_tagname()

Această metodă returnează un tablou unidimensional care conține obiectele de tipul *DomElement* care sunt noduri ale subarborelui al cărui părinte este nodul curent și al căror nume este același cu singurul parametru de tip șir de caractere pe care metoda îl primește ca parametru.

În cazul în care subarborele nu conține un astfel de nod atunci metoda `get_elements_by_tagname()` returnează valoarea logică **FALSE**.

Clasa DomAttribute

Această clasă reprezintă orice atribut pe care un nod din structura arborescentă îl poate avea. Orice nod al arborelui unui document reține două liste de fii. Prima listă conține toate atributele, iar cea de-a doua listă conține restul fiilor.

În continuare vă prezentăm metodele pe care această clasă le introduce pe lângă cele moștenite de la clasa *DomNode*.

Metoda DomAttribute->name()

Metoda `name()` returnează numele atributului curent și are același efect cu apelul metodei `node_name()` moștenită de la clasa de bază *DomNode*.

Metoda DomAttribute->value()

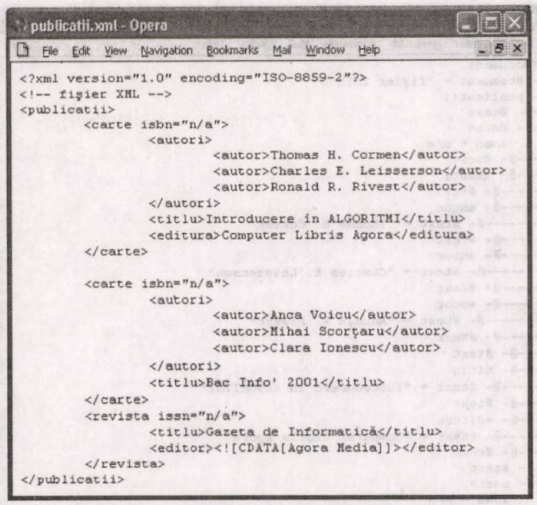
Metoda `value()` returnează numele atributului curent și are același efect cu apelul metodei `value()` moștenită de la clasa de bază *DomNode*.

Metoda DomAttribute->specified()

În cazul în care restricțiile pe care trebuie să le respecte arborele unui document *XML* permit ca un anumit atribut să nu fie specificat, și pentru nodul de tip *DomElement* căruia îi aparține obiectul curent, atributului nu este specificat atunci această metodă returnează valoarea logică **FALSE**. În caz contrar metoda returnează valoarea logică **TRUE**.

Exemple de prelucrare XML

În continuare vă prezentăm câteva exemple de script-uri *PHP* care utilizează fișiere *XML*. Pentru a realiza aceste exemple am folosit fișierul *XML* din figura 1.




```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!-- fisier XML -->
<publicatii>
        <carte isbn="n/a">
                <autori>
                        <autor>Thomas H. Cormen</autor>
                        <autor>Charles E. Leisserson</autor>
                        <autor>Ronald R. Rivest</autor>
                </autori>
                <titlu>Introducere in ALGORITMI</titlu>
                <editura>Computer Libris Agora</editura>
        </carte>

        <carte isbn="n/a">
                <autori>
                        <autor>Anca Voicu</autor>
                        <autor>Mihai Scorţaru</autor>
                        <autor>Clara Ionescu</autor>
                </autori>
                <titlu>Bac Info' 2001</titlu>
        </carte>
        <revista issn="n/a">
                <titlu>Gazeta de Informatică</titlu>
                <editor><![CDATA[Agora Media]]></editor>
        </revista>
</publicatii>
```

Figura 1: Fişierul publicatii.xml

Exemplul 1

Script-ul *PHP* următor ilustrează modul de utilizare a metodei `get_elements_by_tagname()` şi a metodei `get_attribute_node()` aparţinând clasei *DomElement*, iar în figura 2 se poate observa rezultatul execuţiei acestuia.

```
<HTML>
  <HEAD>
    <TITLE>
      Exemplu get_elements_by_tagname()
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
    function main(){
      $dom = domxml_open_file("publicatii.xml");
      if (!$dom){
        echo "Eroare la accesarea fişierului";
        return;
      }
      $root = $dom->document_element();
      $carti = $root->get_elements_by_tagname
                                    ("carte");
      foreach($carti as $carte){
        $titlu = $carte->get_elements_by_
                                tagname("titlu");
        echo "<B>".$titlu[0]->get_content().
                                    "</B><BR>";
        $isbn = $carte->get_attribute_node
                                    ("isbn");
        echo "ISBN: ".$isbn->node_value().
                                    "<BR>";
        $autori = $carte->get_elements_by_
                                tagname("autor");
    if (sizeof($autori)!=0){
     echo "Autori: ";
     for($i=0; $i<sizeof($autori)-1; $i++)
           echo $autori[$i]->get_content().", ";
     echo $autori[$i]->get_content()."<BR>";
```

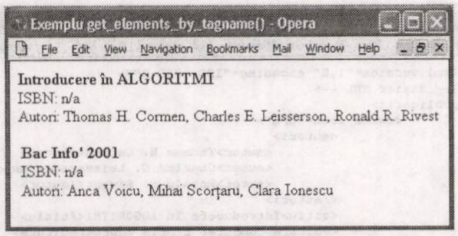


Figura 2: Exemplu get_elements_by_tagname()

```
    }
      }
    }
    main();
  ?>
  </BODY>
</HTML>
```

Exemplul 2

Script-ul *PHP* următor ilustrează modul de utilizare a metodelor al căror prefix este `create_` aparţinând clasei *DomDocument*, a metodei `append_child()` aparţinând clasei *DomNode* şi a metodei `set_attribute()` aparţinând clasei *DomElement*.

În figura 3 se poate observa rezultatul execuţiei *script*-ului.

```
<HTML>
  <HEAD>
    <TITLE>
      Exemplu prelucrare fişier XML
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
    $dom = domxml_open_file("publicatii.xml");
    if (!$dom){
      echo "Eroare la accesarea fişierului";
      return;
    }
    $root = $dom->document_element();
    $revista = $dom->create_element("revista");
    $revista->set_attribute("issn","n/a");
    $titlu = $dom->create_element("titlu");
    $titluc = $dom->create_text_node
                                ("PC Magazine");
    $titlu->append_child($titluc);
    $editor = $dom->create_element("editor");
    $editorc = $dom->create_text_node
                                ("Agora Media");
    $editor->append_child($editorc);
    $revista->append_child($titlu);
    $revista->append_child($editor);
    $root->append_child($revista);
    $s = $dom->dump_mem(false);
    echo "<PRE><CODE>".htmlspecialchars($s).
                                "</CODE></PRE>";
```

Figura 3: Exemplu de prelucrare fişier xml

```
?>
</BODY>
</HTML>
```

Exemplul 3

Script-ul *PHP* următor ilustrează modul de navigare prin structura arborescentă asociată unui document *XML*. Pentru fiecare nod am afişat numele, valoarea şi, unde a fost cazul, atributele. În figura 4 se poate observa rezultatul execuţiei *script*-ului.

```
<HTML>
  <HEAD>
    <TITLE>
      Parcurgere XML recursiv
    </TITLE>
  </HEAD>
  <BODY>
    <?PHP
    function main(){
      $dom = domxml_open_file("publicatii.xml");
      if (!$dom){
        echo "Eroare la accesarea fişierului";
        return;
      }
    echo "<B><PRE><CODE>";
      printRecursive($dom,0);
      echo "</CODE></PRE></B>";
    }

    function printPadding($level){
      for($i; $i<$level; $i++)
        echo "--";
      echo "@- ";
    }

    function printBlanks($level){
      for($i; $i<=$level; $i++)
```

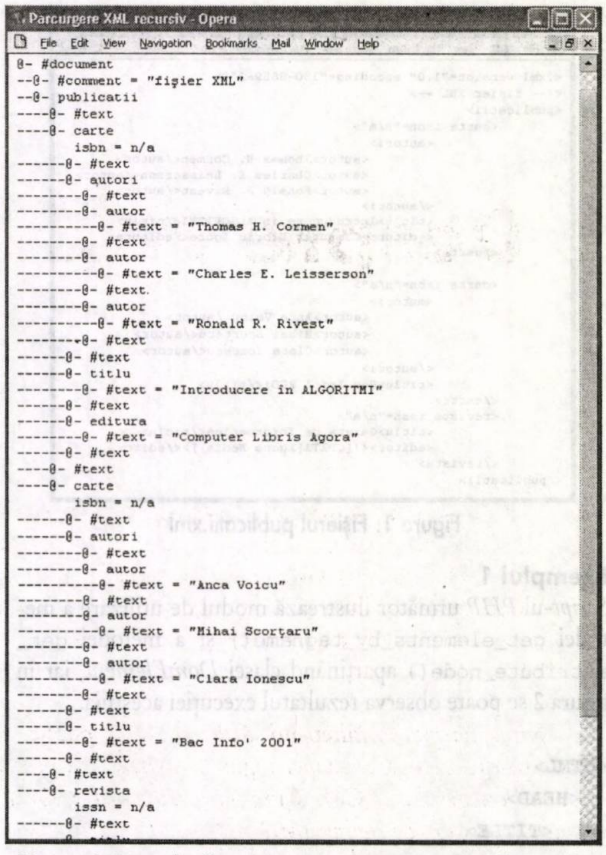


Figura 4: Exemplu de parcurgere recursivă a unui fişier XML

```
        echo "  ";
      echo " ";
    }

    function printRecursive($node, $level){
      printPadding($level);
      echo $node->node_name();
      $s = trim($node->node_value());
      if (($s!="")&&(isset($s)))
        echo " = \"".$s."\"";
      echo "<BR>";
      if ($node->has_attributes()){
        printBlanks($level);
        $attrs = $node->attributes();
        foreach($attrs as $attr)
          echo $attr->name()." = ".
                                $attr->value();
        echo "<BR>";
      }
      $cs = $node->child_nodes();
      foreach($cs as $c)
        printRecursive($c,$level+1);
    }

    main();
  ?>
  </BODY>
</HTML>
```

Despre ACM

Mihai Stroe

O echipă a Universității București a reușit să treacă de faza regională a concursului ACM. Echipa a fost formată din Angel-Stelian Proorocu, Tiberiu Dăneț și Matei Gruber. Am adresat câștigătorilor câteva întrebări referitoare la acest concurs și nu numai. Le mulțumim celor trei pentru amabilitatea cu care au răspuns la aceste întrebări.

Pentru început permiteți-mi să vă felicit pentru câștigarea etapei regionale a concursului ACM. Cum vă simțiți în postura de învingători?

Suntem foarte bucuroși că am reușit să ne calificăm la etapa finală, deși nu ne-am pregătit foarte mult pentru acest concurs. Cu toate că ne fixasem ca scop clasarea în primele zece echipe, datorită unei bune colaborări am reușit să surprindem în mod plăcut.

Ce părere aveți despre modul in care a fost organizat acest concurs?

Organizarea a fost excelentă, deși au fost mici probleme cu sistemul de evaluare *PC*^{2}. Ne-a plăcut în mod special banchetul de final și gustarea oferită în timpul concursului.

Cum vi s-au părut cele opt probleme propuse spre rezolvare?

Problemele au fost abordabile și am reușit performanța de a rezolva șase dintre ele din prima încercare.

Ați fost primii în clasament de la început, după ce ați rezolvat prima problemă, și până în ultima jumătate de oră, când nu s-a mai anunțat evoluția echipelor? Rezultatul final a devenit public, ca de obicei, la festivitatea de premiere. V-a fost teamă de o răsturnare a situației?

Am început concursul cu o problemă mai dificilă, si de aceea după primele două ore eram aproximativ la jumătatea clasamentului. Însă, după aceea am reușit să recuperăm și să ne clasăm pe locul 2 cu jumatate de oră înainte de terminarea concursului, moment din care clasamentul nu a mai fost actualizat. Eram siguri însă că ne vom califica, deoarece "urmăritorii" nostri ne-au mărturisit că nu au reușit să rezolve problema care îi despărțea de noi.

Știu că toți ați participat în timpul liceului la diferite concursuri naționale și internaționale de informatică. Care au fost cele mai importante rezultate pe care le-ați obținut în această perioadă?

Angel-Stelian Proorocu: Personal m-am calificat de patru ori în lotul național obținând în liceu două premii I și două premii 2 la *Olimpiada Națională de Informatică*. Tot în timpul liceului am obținut în anul 1997 o medalie de bronz la *Olimpiada Balcanică de Informatică*. În anul 1998 am obținut o medalie de argint la *Olimpiada de Informatică a Europei Centrale* care s-a desfășurat în *Croația*. În anul 1999 am obținut o medalie de argint la *Olimpiada Balcanică de Informatică* care s-a desfășurat în *Grecia* și o altă medalie de argint în cadrul *Concursului Multidisciplinar Iakutsk*. În anul 2000 am reușit să obțin trei medalii, dintre care o medalie argint la *Olimpiada de Informatică a Europei Centrale* care s-a desfășurat în *România*, o medalie de argint la *Olimpiada Internațională de Informatică* din *China* și o medalie de aur la *Olimpiada Balcanică de Informatică* din *Macedonia*.

Tiberiu Dăneț: Eu am obținut o medalie de bronz la ediția de anul trecut a *Olimpiadei Balcanice de Informatică* desfășurată în *România*. Tot anul trecut am reușit să mă calific în lotul național. Am participat de patru ori la *Olimpiada Națională de Informatică* și am reușit să obțin două premii II, un premiu III și o mențiune.

Matei Gruber: Și eu am obținut o medalie de bronz la ediția de anul trecut a *Olimpiadei Balcanice de Informatică* și, tot anul trecut, am fost membru a lotului național de informatică.

Care sunt, în opinia voastră, principalele diferențe dintre aceste concursuri și ACM?

Vom începe cu diferența evidentă: toate aceste concursuri au fost individuale, iar *ACM* este un concurs pe echipe. În plus, la *ACM* avem dreptul la mai multe încercări pentru o problemă, în timp ce, la celelalte concursuri, se evaluau o singură dată, variantele finale ale soluțiilor. Să nu uităm, însă, că la concursurile la care am participat în timpul liceului puteam obține punctaje parțiale la o problemă chiar dacă, din neatenție sau pentru că uitam anumite cazuri particulare, rezolvarea nu era perfectă. La *ACM* un program trebuie să furnizeze soluții corecte pentru toate seturile de date folosite la evaluare. Ni se pare totuși că la *ACM* se pune un accent mai mare pe partea de implementare decât pe cea de gândire algoritmică.

În majoritatea universităților din România nu se acordă nici o importanță concursului ACM. Studenții afirmă că nimeni nu se ocupă de pregătirea lor. Este diferită situația în cazul vostru?

Din punct de vedere administrativ, lucrurile nu stau prea bine, deoarece *Universitatea* întârzie la limită acordarea celui mai elementar sprijin financiar necesar pentru deplasarea la etapa finală. Din punct de vedere academic însă, cei doi antrenori (*Florin Manea* și *Cristi Grozea*) și-au dat toată silința pentru a ne oferi o cât mai bună pregătire.

Care sunt principalele modalități de pregătire pentru ACM?

Noi ne-am pregătit rezolvând seturile de probleme date la etapele precedente, încercând să simulăm cât mai bine condițiile din timpul concursului. Foarte utile au fost și *site*-urile de genul **www.topcoder.com**.

Credeți că experiența acumulată în timpul liceului a reprezentat un avantaj?

Cu siguranță. Un rol important l-a avut faptul că ne cunoaștem de mult timp și am putut lucra foarte bine împreună.

Până la finală mai sunt câteva luni. Sunteti pregătiți?

Facem tot posibilul să ajungem la *Praga* cât mai bine pregatiți.

Credeți că GInfo poate contribui, în vreun fel, la pregătirea elevilor și studenților pentru concursurile de informatică?

Credem că *GInfo* este o necesitate în pregătirea concursurilor de informatică, dar suntem de părere că există loc de multe îmbunătățiri.

Cine credeți că a contribuit cel mai mult la pregătirea voastră?

Angel-Stelian Proorocu: Eu cred că cel mai important rol în evoluția mea l-a avut primul meu profesor de informatică, *dl. Șcheau Constantin*. Aș vrea să mulțumesc celor care m-au ajutat în timpul liceului să îmi îmbunătățesc performanțele, profesorilor și colegilor din lotul național de informatică.

Tiberiu Dăneț: Țin să le mulțumesc lui *Mihai Stroe* și lui *Radu Berinde* care m-au ajutat enorm în pregătirea pentru olimpiade.

Matei Gruber: Țin să le mulțumesc lui *Angel* și lui *Radu Berinde* care m-au ajutat enorm în pregătirea pentru olimpiade.

Ce planuri de viitor aveți?

Ne înțelegem și lucrăm foarte bine împreună. De aceea am pus bazele unei firme de *software* numită *Think IT*. Sperăm să avem succes în inițiativa noastră, astfel încât să putem sprijini financiar noile generații de olimpici.

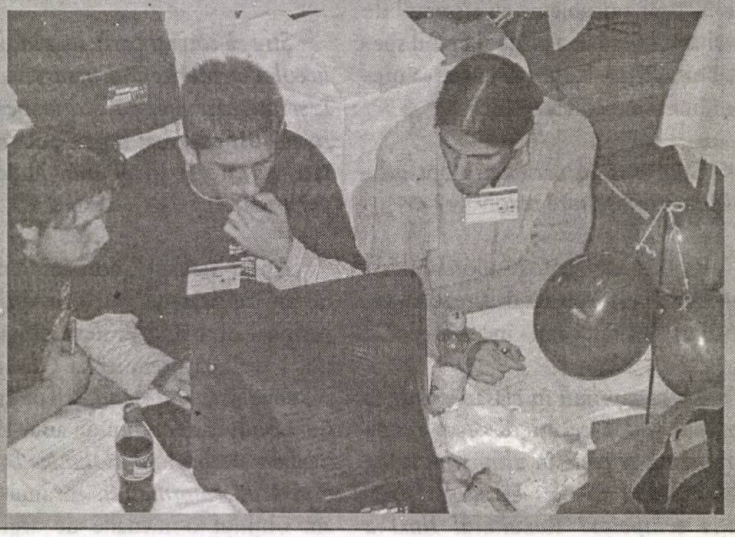
Screenshot...

În imaginea alăturată se află cei trei membri ai echipei cărora li s-a luat interviul anterior.

Această imagine a fost făcută în timpul concursului.

În această imagine se află, de la stânga la dreapta:

Angel-Stelian Proorocu
Matei Gruber
Tiberiu Dăneț.



Cel mai tare ghid IT
cele mai-noi știri
-noi produse
-tari teste
-bune soluții-hard
-soft
-internet
Din aprilie,
citește noul PC Magazine
DELL
Vrei să știi
PC MAGAZINE
ROMÂNIA
www.pcmagazine.ro
SOLUȚII
Tempus
Automatizarea completărilor în Excel
Crearea unui text animat
Despre MIDlet-uri și integrarea cu alte sisteme
INTERNET PRO
Securitatea Rețelei [5]: atacurile DoS
JSP: Java și mai aproape de noi
Transmisia prin fibra optică
GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR
linux
office
grafică
comunicații
LG
Life's Good
Monitor LG
Flatron LCD L1720P
Contract JACK

Publicarea acestei reviste este posibilă datorită companiilor de mai jos care sprijină învățământul românesc de informatică

SPONSORI DE PLATINĂ

FORTE COMPUTERS
Flexibility. Made in Romania

IBM®

s.t
IT SOLUTIONS & SERVICES

hp®

net consulting

FUJITSU COMPUTERS SIEMENS

SPONSOR DE AUR

CCT
CABINETUL CĂLIN TATOMIR

Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-266.516

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

Aprilie 2004
Vol. 14/4

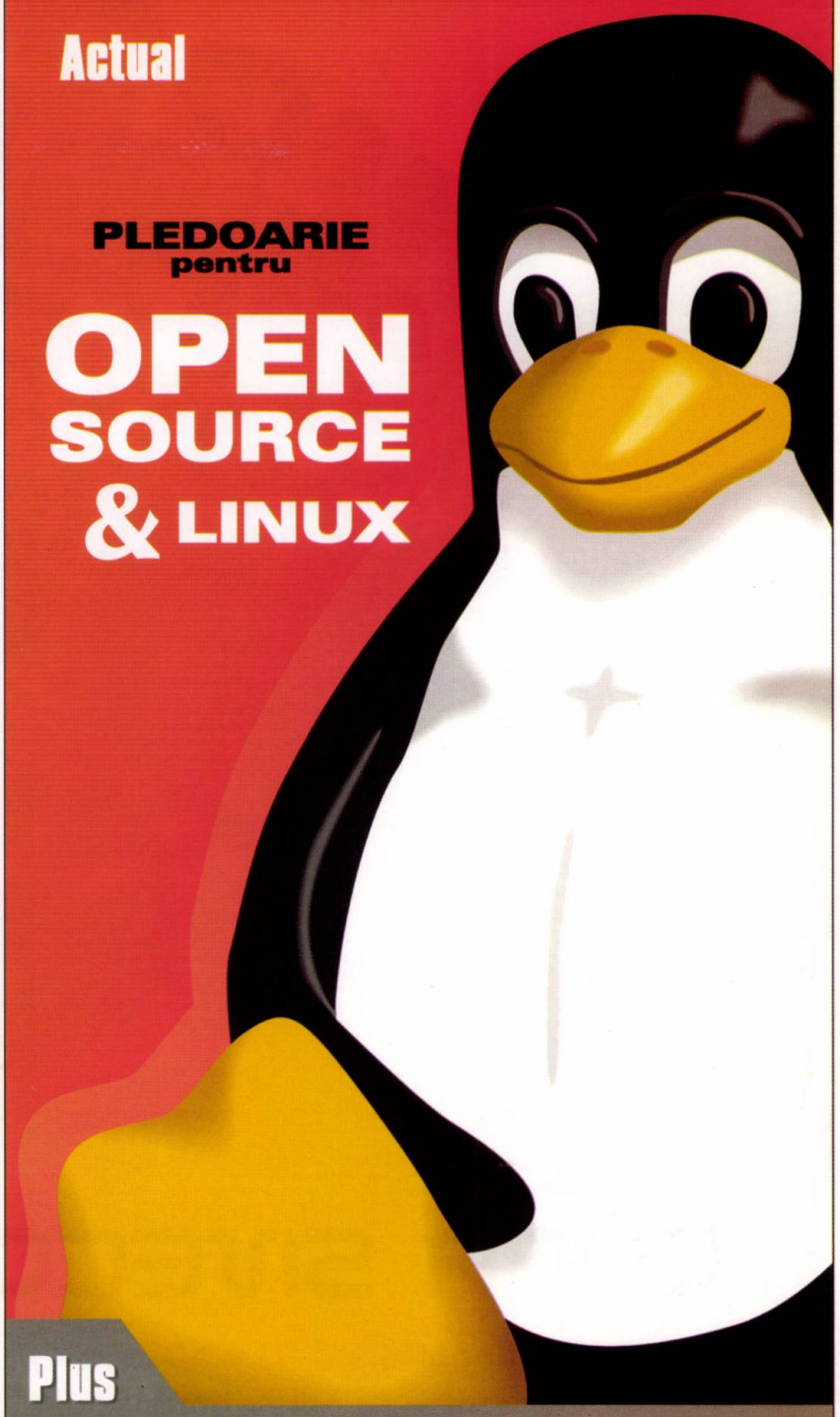
Lei 30.000

din sumarul numărului:

Actual

PLEDOARIE pentru

OPEN SOURCE & LINUX



Plus

probleme
FINALA ACM - Praga, 31 martie 2004

concurs
RUNDELE 26-40·Top 50 după 25 runde

actual
EUROPRIX-concurs european multimedia

5-8 iunie 2004

CUPA SIVECO

Concurs național
de software educațional

Sub patronajul Ministerului Educației, Cercetării și Tineretului
Partener media: PC Magazine România

trimiterea lucrărilor - până la 15 mai 2004

SIVECO ROMANIA SA

PC MAGAZINE ROMÂNIA

MINISTERUL EDUCAȚIEI,
CERCETĂRII ȘI TINERETULUI

Vol. 14, Nr. 4 /2004

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Director Bursele Agora:
Mihai Scorțaru
Redactor-șef:
Claudiu Soroiu
Redactor-șef adjunct:
Carmen Șușman
Redactori:
prof. *Eugen Ionescu*
Mihai Stroe
Colaboratori:
Cristian Olar

Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
400091, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/592.422, 0265/266.516
Fax: 0265/266.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359
Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

Director general:
Romulus Maier
Director executiv:
Adrian Pop

Vânzare reclamă:
Mihai Bucuroiu
Denis Marioara
Marius Comărniceanu
Difuzare:
Florin Dinu
Florin Oprescu
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Σ Sumar

actual.. PAG. 5

Pledoarie pentru OPEN-SOURCE și LINUX

Cristian OLAR

~ Windows vs. ~
~ Open-Source ~
~ și Linux ~

EUROPRIX

~ concurs european ~
~ multimedia pentru ~
~ tinerii talentați ~

probleme.. PAG. 10

Finala ACM

~ 31 Martie 2004~
~ Praga ~

Olimpiada NAȚIONALĂ de informatică 2004

~ enunțurile celor 18 ~
~ probleme propuse spre ~
~ rezolvare elevilor ~
~ de liceu ~

soluții.. PAG. 31

Faza JUDEȚEANĂ a olimpiadei de INFORMATICĂ pentru LICEE

Mihai STROE

concurs.. PAG. 34

Despre rundele #26-#40

Top 50 după 40 runde

Rundele #41-#50

internet.. PAG. 44

Pagini WEB cu PHP 4

Mihai SCORȚARU
Claudiu SOROIU

~ introspecție ~
~ și reflexie în PHP ~

GInfo încurajează publicarea de articole de către cititorii săi; în fiecare număr apar articole care nu sunt scrise de membrii redacției.

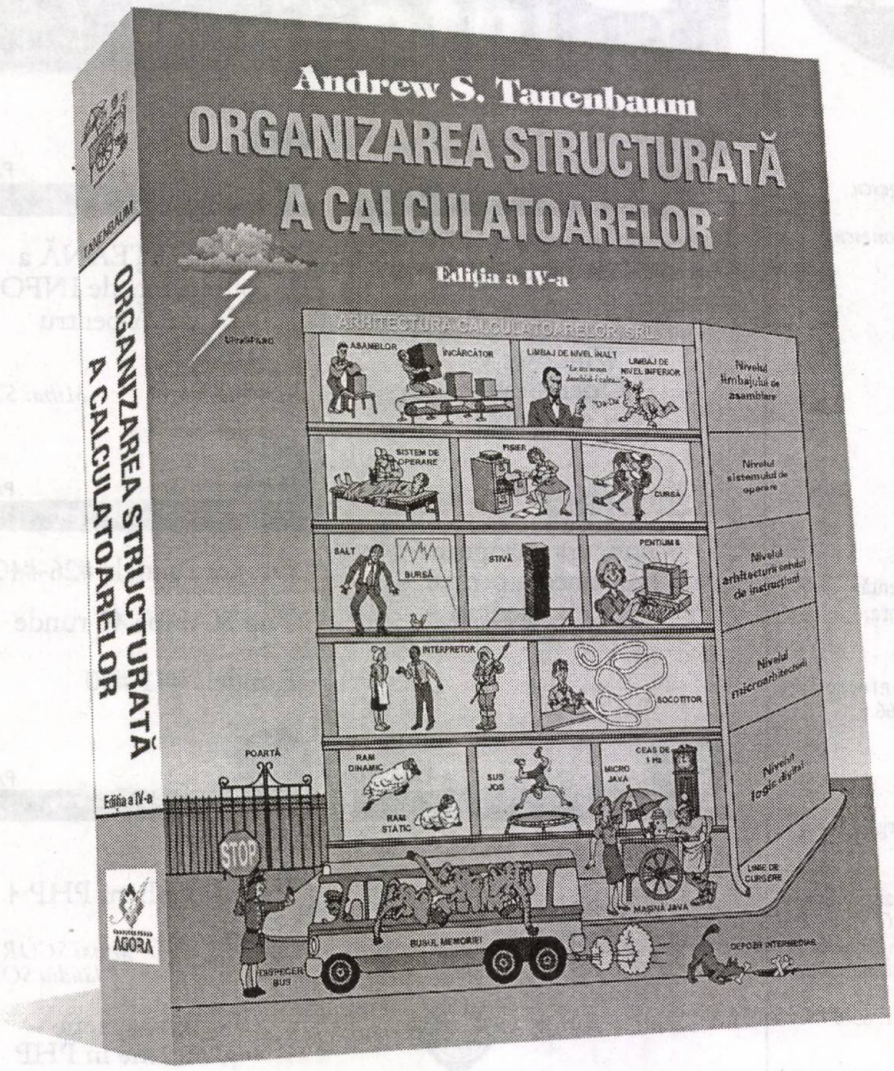
Dacă doriți ca numele dumneavoastră să fie inclus pe lista colaboratorilor revistei noastre, tot ce trebuie să faceți este să ne trimiteți un articol în vederea publicării.

Singura condiție pe care trebuie să o îndeplinească articolul este ca tema să se încadreze în profilul revistei noastre.

Este recomandabil ca nivelul de accesibilitate să fie unul mediu și înțelegerea informațiilor prezentate să nu implice deținerea de cunoștințe avansate din alte domenii. Pentru mai multe detalii vizitați:
http://www.ginfo.ro/revista/publicare.shtml

Cine are carte, are parte!

...de înțelegerea calculatoarelor!



Comandă-o acum și dacă ești student

primești și o **REDUCERE** semnificativă!

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65, 0721-927.499

TENDINȚĂ sau pur și simplu FENOMEN?

Din nou vă voi împărtăși ideile mele legate de "cuvinte": cuvinte scrise, cuvinte rostite, modul nostru de a ne exprima, de a comunica. Aproape mă bucur că nu voi apuca să văd ce se va întâmpla în anii YZ (cât să scriu aici? mă tem că anul YZ nici nu este așa de departe). Dar nu v-am spus care este problema concretă de care m-am împiedicat (de fapt pentru a nu știu câta oară).

Am avut plăcerea să particip la câteva conferințe de specialitate, susținute de profesioniști în tehnologia informației și comunicației. Oameni tineri, entuziaști, foarte bine pregătiți, mulți dintre ei cu talente incontestabile în a vorbi, în a explica, în a ține o prelegere. Expuneri subliniate cu prezentare de *slide*-uri în *PowerPoint* realizate cu pricepere, respectând toate regulile meseriei. Până aici toate bune și frumoase. Dar ...

La un moment dat, unul dintre vorbitori s-a prezentat rostindu-și numele, apoi adăugând: *sunt sisadmin*. Am fost atentă în continuare și am constatat dezamăgită că nu se vorbește *românește*. Dar nu se vorbește nici în engleză. Se folosește un amestec ciudat a celor două vocabulare, întrețesute cu termeni de specialitate rostite fie fonetic, fie după regulile limbii engleze în care sunt mai multe abrevieri decât cuvinte întregi.

M-am uitat în jur. Nimeni nu părea mirat. Nimeni nu dădea semne că nu ar înțelege. Da. Și eu am înțeles. Chiar și pe cele auzite acum prima oară. Întrebarea pe care o pun este totuși următoarea: de ce acest fenomen? Ce câștigăm din faptul că în loc să spunem *utilizator* spunem *user*, în loc să spunem *afacere*, spunem *business*? Cel din urmă exemplu diferă oarecum de primul: cuvântul *business* face parte din vocabularul nostru mai de mult. S-a furișat printre cuvintele folosite în viața de zi cu zi nu doar în cazul specialiștilor *IT* (rostit "aiti"). *IT* constituie un alt exemplu. În folosirea acestei prescurtări există totuși un strop de rațiune: este mult mai scurtă decât descrierea în clar în limba română: *tehnologia informației*. Dar ce părere aveți de cuvântul *agreement*? De ce nu am putea folosi *acord*-ul din limba română? Are doar două silabe! Parcă aud cum cineva se grăbește să mă contrazică afirmând că cele două cuvinte nu au același sens, sau cel puțin există diferențe în nuanțe. Am convingerea că nuanța ar apărea oricum din context...

Apar atunci întrebările: suntem chiar așa de grăbiți? Suntem chiar așa de presați de criza de timp? Sau de multe ori suntem doar comozi? Majoritatea persoanelor care își "colorează" vocabularul cu astfel de termeni împrumutați din limba engleză, chiar dacă par mereu grăbiți, nu sunt deloc comozi. Dimpotrivă, sunt foarte ambițioși și învață în continuu. Atunci? Folosind termeni originali din limba engleză suntem tentați să credem că suntem *mai credibili*. Credem că folosind sursa însăși, fără să apelăm la traducere, informația pare mai sigură. Traducerea presupune deja "interpretare", a intervenit factorul uman, deci "cine știe cum e de fapt?" Cred că ne este frică de cuvinte care în mod consacrat în limba noastră maternă au fost și sunt folosite în contextul unor activități umane: cum ar suna să spunem că programul a fost *tradus* în loc de *compilat*? Probabil mai există încăpățânați care chiar așa se exprimă.

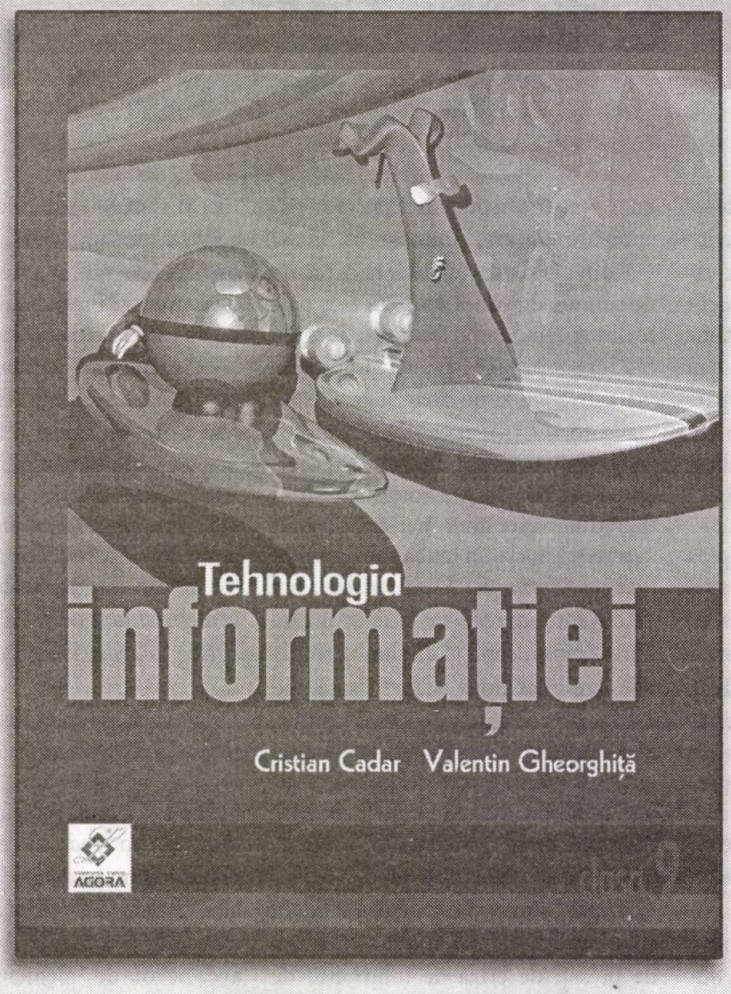
Avalanșa de termeni noi, apărută și răspândită în ultimul deceniu ne-a luat prin surprindere. Într-adevăr nu mai este timp să căutăm corespondențele termenilor *hard*, *soft*, *desktop*, *server*, *link*, *browser*, *designer* sau să ne "muncim" să spunem *pagină Web* în loc de *site*, *conducere* în loc de *management*, *studiu de piață* în loc de *marketing*, *diapozitiv* în loc de *slide* etc.

Revin la problema conceptuală inițială. Ce se va întâmpla în viitor? Deocamdată ne înțelegem. Nu se știe dacă medicii sau actorii (sau...) ne-ar înțelege, dar nu cred că acest aspect interesează acum pe cineva. Întrebarea este alta: ce câștigăm prin această tendință? (Sau este fenomen?) Azi-mâine vom avea nevoie de "memento la purtător" pentru a ne descurca printre abrevieri, sau va trebui să ne mulțumim cu *poteca noastră îngustă* unde ne descurcăm, restul rămânând "problema altora".

Dar chiar și până atunci, nu văd de ce nu putem spune a *personaliza* în loc de *customiza*, *tabel* în loc de *teiběl*, *memorie* în loc de *storigi* etc. Altfel, va trebui *să ne facem upgrade și update, să realizăm daily reports, dar să facem și lobby și monitoring pentru settingsurile noastre, după care să lansăm un load balancing pentru a evita crashul creierului propriu (sau a storigiului intern?)*.

Clara Ionescu

COMPUTER LIBRIS AGORA



Lucrarea de față își propune să pregătească cititorul în scopul utilizării unui calculator personal. Ea urmărește programa școlară, aprobată de Ministerul Educației Naționale, pentru disciplina "Tehnologia informației", disciplină în planul de învățământ a claselor a IX-a la profilul teoretic și pedagogic. Manualul corespunde și programei de la disciplina "Informatică - Tehnologii asistate pe calculator", profilul tehnologic.

Pentru parcurgerea acestei cărți nu sunt necesare cunoștințe prealabile în domeniu, termenii și noțiunile prezentate sunt explicate detaliat în carte. Exemplele sunt sugestive, cartea nu este încărcată inutil cu informații care depășesc programa amintită.

COMPUTER LIBRIS AGORA

COMPUTER LIBRIS AGORA
Informații la tel. 0264 - 592422

Pledoarie pentru OpenSource și Linux

Cristian Olar

Trăim vremea pledoariilor? Ne bucurăm să constatăm că pledoaria pentru Java a stârnit interes. Mulți cititori de-ai revistei ne-au căutat cu părerile lor, dintre care unele le-am și publicat. Dar la fel ne-am bucurat, citind pledoaria de mai jos. Gând la gând cu bucurie...

Am citit articolul *Pledoarie pentru Java* cu foarte mult interes. Pe de o parte pentru că părea ceva revoluționar, pe de altă parte pentru că vara trecută am experimentat și eu puțin limbajul respectiv, dar numai așa, pe la nivelul genunchiului broaștei. Mi s-a părut interesant, însă nu am avut prea mult timp să adâncesc subiectul pentru că urma clasa a XII-a și am avut altele de făcut, iar în timpul liber acordat experimentelor m-am ocupat de o nouă pasiune de-a mea: modelări *OpenGL* (și aici fiind tot pe la început, deși ceva mai avansat decât eram la *Java*).

Ceea ce mi-a atras atenția imediat a fost propunerea din *GInfo* de a folosi *soft*-uri *OpenSource* în școli în locul compilatoarelor care teoretic ar cam trebui să coste. Eram foarte atras de idee, mai ales pentru că de aproximativ 10 luni am făcut trecerea spre *Linux* pe care acum îl folosesc aproape exclusiv. A doua zi după apariția articolului am mers foarte nerăbdător să discut chestiunile din articol cu profesoara mea de informatică știind că și dumneaei citește revista. Am fost dezamăgit să văd că și aici articolul fusese judecat ca "o critică aspră la adresa profesorilor de informatică". Bineînțeles că simțise și alte aspecte ale problemei, dar acestea au fost cele care au atins-o mai tare. În luna următoare, citind reacțiile cititorilor la articol, am observat destul de dezamăgit că toți profesorii care au citit articolul și au trimis un răspuns au fost atinși tot de aceeași parte a sa. Atunci am luat frumos revista și m-am uitat la ea. Dacă luăm articolul de trei pagini și ceva și măsurăm din ochi rândurile referitoare la profesori, vom observa că acestea reprezintă aproximativ un sfert de pagină, adică aproximativ 8,3% din spațiul articolului. Oare cititorii chiar nu și-au dat seama că era vorba doar despre o "paranteză"? Poate puțin cam aspră, dar nu aceasta era esența articolului.

Mă voi întoarce la *OpenSource*. În școli la disciplina informatică se studiază în continuare sisteme de operare care costă, se folosesc *soft*-uri care costă, și încă nu oricât, ci chiar foarte mult. Lucrurile însă nu stau așa din cauza pirateriei în floare din *România* dar, așa cum s-a specificat în articol "lucrurile nu vor sta întotdeauna așa". La un moment dat românii vor realiza că așa nu se mai poate, dacă vrem să ne integrăm în lumea dezvoltată și nu în lumea a III-a. Dar nici atunci nu ne vom îmbogăți peste noapte. Imaginați-vă că cineva, în acest viitor, dorește să cumpere un calculator. Asta încă și-o permit mai mulți români. Să spunem că plătește 800 $ pentru un calculator nou nouț. Bun, și acum ce face, mai plătește încă atât pentru o licență *Windows*, un *Office* și, Doamne ferește, dacă este inginer mai trebuie să plătească de zece ori atât pentru o licență *AutoCAD*? Soluția se află în fața noastră și nu trebuie să o ignorăm: *OpenSource*. Pentru persoane fizice s-ar mai putea trece peste aceste dificultăți, în schimb școlile nu ar avea posibilitatea să cumpere toate programele necesare în procesul de învățare. Trecerea pe sisteme *Linux* nu ar necesita nici un ban, doar timpul de instalare care nu este mai lung decât cel de instalare al *Windows*-ului. De fapt educația nu ar trebui să fie blocată de lipsa de fonduri. Dacă există potențial în țara asta, atunci de ce să ne lăsăm împiedicați de asta mai ales când cauza piedicii este o simplă prejudecată pe care am auzit-o de mai multe ori: "*Linux-ul nu este user-friendly...*". Asta o cam spune toată lumea fără ca măcar să vadă despre ce este vorba. Recent am dus la școală distribuția *Linux Knoppix* 3.3. Pot să spun că cei doi-trei colegi interesați care au aruncat o privire au fost pur și simplu încântați. Interesată a fost și domnișoara dirigintă, dar nu știu câți alți profesori ar fi fost la fel. Reacția colegilor m-a determinat să scriu această

scrisoare care inițial plănuiam să o trimit ministerului învățământului la vreo adresă de relații cu publicul, însă nu am găsit niciunde așa ceva pe **www.edu.ro**, deci m-am hotărât să scriu la *GInfo* unde cred că voi fi ascultat, mai ales pentru că un subiect asemănător a fost deja deschis.

Așadar, de ce să trecem pe *Linux*?

1. Prorgamarea se face în același fel în care s-a făcut pe *Windows*, cu unele avantaje. *Pascalul* este tot *Pascal*, *C++* este tot *C++*, *Java* este tot *Java* și lista poate să continue. În plus, toate mediile de dezvoltare sunt gratuite și au facilități de *debugging*, ca și mediile de dezvoltare tradiționale folosite în școli (*Borland Pascal*, *Turbo Pascal*). Pe lângă toate acestea, existența lui *DosEmu* permite celor care și-au compilat programele acasă, folosind compilatoarele tradiționale, o prezentare fără dificultăți pe *Linux*, asta dacă din anumite motive respectivul elev sau profesor nu dorește ca sursa programului să fie cunoscută. În caz contrar programul poate fi recompilat pe *Linux* fără mari modificări. Celor care folosesc *Delphi* sau *Visual Basic* 6 sau *.NET* li se aplică același tratament datorită lui *WINE*, amândouă pachetele amintite fiind gratuite și putându-se descărca fără dificultăți de pe *Intenet*, fiind în total sub 50 MB. Chiar dacă nu vă vine să credeți, pentru *Visual Basic* 6 există posibilitatea de editare a fișierelor sursă cu medii de dezvoltare ca și *Anjuta IDE*, deși despre un compilator care să funcționeze pe *Linux* pentru *VB6* nu am auzit, deși, repet, programele gata compilate vor rula impecabil cu *WINE* și chiar mai stabil.

2. Stabilitatea sistemelor UNIX (Linux) constituie caracteristica pentru care sunt cunoscute. Sunt sigur că multor programatori li se întâmplă să folosească pointeri aiurea din cauza unei neatenții în fazele inițiale ale scrierii unui program. În *Windows*, asta de obicei, dacă nu cumva ai noroc ceresc, îți va bloca sistemul și te va trece printr-un proces enervant de restart până la posibilitatea de revenire asupra problemei. *Linux*-ul nu permite acest lucru, chiar dacă ai alocat greșit vreun pointer. Programarea se face de-a dreptul relaxant, fără nici o bătaie de cap din partea sistemului de operare.

3. Se constată prezența crescândă a numărului de dezvoltatori ai sistemului. Linux este *OpenSource* și asta implică testarea lui de milioane, poate chiar miliarde de utilizatori înainte de eliberarea sistemului în varianta finală. Practic, avem un sistem care a rulat pe alte milioane de configurații. Posibilitatea ca PC-ul nostru să fie cel de-al 1000000001-lea și să apară conflicte între componente este mult mai mică decât la oricare alt sistem.

4. Gratuitatea. Da, *Linux* este complet gratuit și nimeni nu îl va folosi vreodată pentru a crea monopoluri. Nu vă va obliga să plătiți încă jumătate din prețul calculatorului dumneavoastră pentru a putea realiza ceva efectiv cu el.

5. Configurabilitatea. Un element care nu va crea probleme începătorilor, deoarece, de obicei, există o configurație inițială care se potrivește majorității sistemelor de calcul, dar, în același timp aduce un mare avantaj pentru avansați care își pot configura totul în *Linux*, până chiar și felul în care procesorul prelucrează informațiile.

6. Gama mare de aplicații prezente pentru Linux. Pe *Internet* există foarte multe aplicații care pot fi mult mai ușor de găsit decât în cazul celor *freeware* pentru *Windows*, deoarece se găsesc toate pe *site*-uri ca **http://www.sourceforge.net**, **http://www.freshmeat.net** sau **http://www.happypenguin.org**. Și aici mă refer inclusiv la programe educative. Sunt acoperite toate capitolele prezentate pe *AEL* și îndrăznesc să spun că sunt realizate mai bine (temă de curiozitate: intrați pe **www.google.com** și lansați o căutare pentru cuvântul cheie *idiotware*, notați primul rezultat). Veți obține, de exemplu, *Celestia* pentru astronomie și suita de aplicații de *Edutainment* din *KDE*.

7. Modul de rulare al Linux-ului. Mă refer la sistemul *multi-user* care permite rularea mai multor aplicații fără ca măcar să părăsim modul de lucru în consolă, sau la posibilitatea că, pe lângă faptul că avem la dispoziție un mediu *multi-tasking*, acesta are la rândul lui mai multe *desktop-uri* virtuale, pe care le accesăm printr-un simplu click. Șase *window manager*-e, printre care și *fwm*95, care, deși este destul de slăbuț în comparație cu *GNOME*, *KDE* sau *XFCE* (toate fiind denumiri de *window manager*-e), implementează mediul de lucru al *Windows*-ului.

8. Coexistența pașnică între Windows și Linux. Trebuie să fim conștienți că trecerea de la unul la altul nu poate avea loc peste noapte. Adaptarea noastră la *Linux* necesită un oarecarea timp care, după părerea mea, poate fi redus la 2-3 luni. Este vorba despre faptul că, inițial nu știi care aplicație ce face: dintr-o dată în loc de *WinZip* ai *FileRoller*, în loc de *Adobe Acrobat Reader* ai *X PDF Viewer*, în loc de *Explorer* ai *Nautilus*, în loc de *Internet Explorer* ai *Mozilla Firefox* sau *Netscape* etc. În momentul când vrei să faci ceva repede, vei apela probabil la *Windows*. Oricum, adaptarea nu se face în prea mult timp.

9. Suite de Office compatibile cu MSOffice. Eu folosesc *OpenOffice* pe care cei din familia mea îl folosesc sub *Windows*, însă există și alte variante. La fel ca și alte proiecte *OpenSource*, *OpenOffice* este gratuit și în scurt timp se va produce și varianta în limba română. Dicționare de corectare la tastare există deja. Toate funcțiile din *MSOffice* sunt implementate și aici, existând chiar și altele în plus.

10. X Window System, (interfața grafică a Linux-ului) este considerată implicită de distribuții noi (ca *Fedora Core* 1). Când aceasta rulează comenzi din *Windows* (pentru crearea unui director, a unei scurtături ori a unui document) sunt aceleași ca și în *Windows* (click dreapta pe *desktop* sau

în vreun *browser* și alegere din meniul *popup*). Din curiozitate mi-am propus să rezolv subiectele de Atestat (*Sisteme de Operare*) propuse în județul Arad la finalul clasei a XII-a pentru clasele de informatică. Vreau să vă spun că în afară de cerința cu realizarea dischetei sistem pentru care a trebuit să rulez o aplicație, toate celelalte s-au realizat cu exact aceleași mișcări pe care le-aș fi făcut în *Windows*.

11. Jocurile. Nu am să insist asupra motivului, dar știu că pentru mulți dintre utilizatori sunt importante și jocurile. Cele de pe **www.happypenguin.org** sunt destul de plăcute pentru cei care doriți doar să vă relaxați după rezolvarea unei probleme de olimpiadă care v-a consumat. Însă pentru cei care sunt pasionați de jocuri ca și *Counter Strike*, *Morrowind*, *Warcraft III* și altele vă spun doar că *WineX*, o ramură a *Wine*, specializată pe implementarea *DirectX API*-ului sub *Linux*, folosind *OpenGL* (metoda de accelerare grafică folosită de *Linux*), le poate rula pe toate acestea. Unele le-am testat și eu. Colegii îmi spuneau acum doi ani că ei nu-și instalează *Windows XP* pentru că un cutare joc nu se rulează pe sistemul acesta. Ei, aveam și eu un joc de genul respectiv care pe *Linux*, folosind *WineX* rulează cu bătăi de cap...

12. Multimedia. *Linux*, cu doar două pachete de *codec*-uri, numite *w32codec* și *mpeg321* rulează orice format audio sau video de la *mp3* până la *ogg*, *asf*, *wma* sau *wmv*. Playere-le *MPlayer* sau *Xine* suportă ambele subtitrări și vor vedea și *DVD*-urile. Claritatea imaginii este extraordinară și deși nu mi-am mai făcut update la *w32codec* de aproape jumătate de an, toate filmele rulează, pe când utilizatorii de *Windows* au fost nevoiți să facă update la *DivX* și *Xvid* cel puțin de două ori în aceeași perioadă.

13. Internet. Prin însăși definiția lui, *Linux* este legat de *Internet*. Îndrăznesc să spun că fără *Intenet Linux* nu ar fi luat ființă. Există o mulțime de *browser*-e pentru *Internet* cum ar fi *Netscape*, *Mozilla*, *Mozilla Firefox*, *Galeon*, *Konqueror Browser* și altele care pot fi folosite fără probleme. Există varianta de *Yahoo Messenger* pentru *Linux*, chiar pe site-ul *Yahoo*. *Linux* are și propriul său serviciu de *Messenger*. *Xchat* înlocuiește cu mult succes (după părerea mea) *mIRC*-ul din *Windows* deși, acesta poate fi rulat cu ajutorul *WINE* și pe *Linux*. Există multe, multe forum-uri pentru întrebări și asistență tehnică gratuite pentru *Linux* pe *Internet*.

14. Grafica pe calculator. După cum am amintit deja, *Linux* este accelerat grafic, doar că *Linux*-ul folosește *OpenGL* și nu *DirectX*. Am încercat de-a lungul timpului ambele tehnici de programare și sunt de părere că *OpenGL* este mai interesant, dar aici fiecare hotărăște singur. Oricum, atât *ATI*, cât și *nVidia*, precum și o serie de alte firme au produs *drivere* pe *Linux* pentru plăcile lor grafice. Zvonurile de pe *Internet* spun că Episodul 3 din *Star Wars* a fost completat cu efecte speciale lucrându-se exclusiv sub *Linux* pentru că cei de la efecte speciale se săturaseră de timpii mari de renderizare și de montare a filmelor sub *Windows*. Nu știu cât este de adevărat. Cert este că toate programelele mele cu *OpenGL* rulează vizibil mai ușor pe *Linux* decât pe *Windows*. Apoi, *Linux*-ul mai are *GIMP* (*GNU Image Manipulation Program*) care este (din nou: după parerea mea) cel mai performant în domeniu (cine dorește, poate să mă contrazică aici pentru că este adevărat că în afară de *Adobe Photoshop*, *Ulead PhotoImpact*, *Fireworks* și *GIMP* altceva nu cunosc).

15. Portabilitate. Cred că este un aspect foarte important al problemei. Câteodată va trebui să ne adaptăm puțin programele pentru a le putea compila pe alte sisteme de operare. Și acest "puțin" trebuie să fie cât mai puțin. Din fericire *Linux* excelează și la acest capitol. Programarea se face în *Linux*, de obicei, folosind librării care au variante atât pentru *Linux*, cât și pentru *Windows*, *MacOS*, *FreeBSD*, *UNIX*, *BeOS*, *OS/2*. În concluzie, programele vor putea fi portate nu doar pe *Windows*, ci pe o mare gamă de alte platforme. Aici s-ar putea să spuneți: "Bine, bine dar dacă tot sunt portabile, atunci de ce să nu folosim librăriile respective cu varianta pentru *Windows* și să portăm de pe *Windows* pe celelalte platforme?". Puteți într-adevăr să faceți și așa, însă mediile de dezvoltare din *Linux* vă oferă facilități atunci când folosiți aceste librării, cum ar fi compilări optimizate sau *IntelliSense Input* pentru ele, facilități pe care mediile de dezvoltare din *Windows* nu vi le vor oferi (decât eventual *IntelliSense*-ul), însă ar mai fi de considerat pe lângă asta și motivul 2...

Am ales 15 motive, cu toate că multe altele ar mai putea fi date. Nu știu dacă sunt cele mai importante, dar este un început pentru deschiderea unui dialog și sper că, deși, nu am reușit să trimit e-mail-ul acesta ministerului, propunerile mele nu vor fi ignorate.

Încă un lucru important pe care l-am citit în articolul *Pledoarie pentru Java* a fost acela că sistemul educațional uită un lucru foarte important: un limbaj de programare (în cazul de față un sistem de operare) nu este "mai greu" pentru un elev proaspăt, decât celălalt, din simplul motiv că acesta inițial nu știe nimic. Profesorii sunt cei care trebuie să-l formeze și dacă este format pe *Linux*, puteți fi siguri că *Windows*-ul i s-ar părea ciudat și nu "user friendly".

Vă citez un mesaj de pe *Internet*:

Linux is not user-friendly.

Dar, există și răspuns, tot pe *Internet*:

Linux _IS_ user-friendly, it is not idiot-friendly or ignorant-friendly!

Așa că nu avem de ce să ne facem griji.

Cristian Olar este elev în clasa a XII-a. El poate fi contactat prin e-mail la adresa **ogavril@rdslink.ro**.

Concurs!

EUROPRIX

Începând de luna aceasta EUROPRIX este gazda celei de-a șaptea ediții a unuia dintre cele mai importante concursuri europene — Top Talent Award — care are ca subiect dezvoltarea de aplicații din domeniul IT.

Ediția 2004 a concursului *TTA* (*Top Talent Award – Premiile celor mai talentați*) va avea loc la *Salzburg* și se adresează tuturor tinerilor din *Europa* care au cel mult 30 de ani.

Acest concurs este un eveniment cheie al *Europei* și se adresează tinerilor pasionați de conținuturi multimedia. În cel de-al șaptelea an al existenței sale, proiectul demarat de *Europrix* atrage cu el noutăți ca *Top Talent Content Fusion Award* (*Premiul pentru cele mai bune conținuturi multimedia interconectate*) pe tema *Muzica întâlnește Multimedia* și pentru prima dată *Thesis Award* (*Premiul pentru cea mai bună teză*) care se acordă pentru cele mai bune idei științifice ale studenților europeni în domeniul multimedia.

Participanții la acest concurs pot prezenta proiecte inovative și conținuturi multimedia creative care pot folosi orice platforme și canale care suportă multimedia:

- *Aplicații on-line*: Orice conținut *web* și aplicații, în special cele care indică potențialul experienței pe scară largă care demonstrează o interactivitate ridicată, interconectare globală, transmisiuni în timp real de conținut multimedia.
- *Offline – DVD*: Proiecte clasice și de viitor care pun un accent puternic pe modul de a ilustra povestiri într-o manieră emoționantă, expresivă și cu mult conținut multimedia.
- *Conținuturi mobile*: Conținuturi inovative și aplicații care utilizează la maxim potențialul dispozitivelor compacte, mobile, pe care comunicarea se realizează intens și care să se concentreze pe soluții multimedia de actualitate.
- *Conținuturi TV și Video*: Conținuturi pentru cele mai noi tipuri de transmisiuni ale viitorului digitizat care vizează în mod direct utilizatorii acestor tipuri de sisteme.
- *Platforme pentru jocuri*: Conținuturi care, prin interacțiuni de mare viteză în care utilizatorul are cel mai important rol, să-l țină în suspans.
- *Grafică pe calculator interactivă*: Conținuturi vizuale explozive care transformă realitatea și care introduc utilizatorii în lumea virtuală a naratorilor cibernetici.
- *Unelte pentru conținuturi multimedia și pentru proiectarea interfețelor*: Unelte puternice pentru prelucrarea și manipularea conținuturilor multimedia și inovații tehnologice care implică noi modalități de proiectare a interfețelor.
- *Cross-Media*: Proiecte care încurajează utilizarea conținuturilor multimedia pentru o mai mare flexibilitate și utilizabilitate.

Cele mai bune proiecte vor fi premiate cu 10.000 de euro fiecare.

În afară de aceste primii, în comparație cu ediția de anul trecut a concursului, pe lângă cele opt categorii care acoperă întreaga scenă multimedia cu toate ramurile sale, ediția din acest an introduce două noi premii *The Top Talent Content Fusion Award* și *The TTA Thesis Award.*

Noutatea nr. 1 — Multimedia și muzică

În acest an premiile *EUROPRIX* pentru cei mai talentați îi încurajează pe toți tinerii producători europeni să participe cu produsele și aplicațiile lor, în special cele care folosesc componente audio. Toate aceste aplicații și produse vor fi jurizate din punctul de vedere al legăturii dintre muzică și multimedia. Premiul nu va fi acordat numai dezvoltatorilor actuali din domeniile tehno, house și hip-hop, ci va căuta și cele mai bune soluții care includ compozitori clasici sau muzică jazz, prezintă instituții sau oferă soluții de ultimă oră în domeniul comerțului cu muzică.

Industria de conținuturi și produse multimedia este în continuare pe primul plan. Premiul pentru cele mai bune conținuturi multimedia interconectate dorește să se extindă și în noile domenii audio care captează emoțional. Așadar, utilizatorii pot pătrunde în lumile create de multimedia, iar premiul dorește să încurajeze acest domeniu, afirmă *dl. prof. Peter A. Bruck* inițiatorul concursului.

Noutatea nr. 2 — Cercetări multimedia

Pentru prima dată în istoria sa, concursul *EUROPRIX* se adresează tinerilor elevi și studenți ai universităților, co-

legiilor sau școlilor cu profil orientat spre multimedia. Această secțiune se concentrează în jurul experienței practice a tinerilor. Așadar, în acest an acest concurs va fi discutat și promovat în mod special în majoritatea școlilor și universităților orientate spre multimedia, din toată *Europa*.

Se va pune accent pe conținutul științific și se vor analiza următoarele titluri:

- *Tinere staruri multimedia: povestiri de succes și cariere paradigmatice*;
- *Studii de caz ale viitoarelor companii multimedia*;
- *Produse multimedia cu succese pe plan economic*;
- *Planuri de afaceri multimedia de succes sau ineficiente*;
- *Criza sau renașterea unei noi economii?*
- *Întrebări despre convergențele și divergențele tehnologiei, economiei și artelor*;
- *Prototipuri inovatoare pentru industrie*;
- *Probleme cu licențele în domeniul afacerilor multimedia*;
- *Tinerele talente și industria creativă*.

Celor mai bune trei teze vor fi premiate cu 1000 de euro fiecare la categoria *Premiul pentru cea mai bună teză*.

Programul concursului

Anul trecut au participat la acest concurs 425 de tineri din *Europa* reprezentanți ai întregului spectru contemporan al culturii și stilurilor multimedia care au completat cel de-al șaselea an al istoriei competiției *TTA*.

Cele mai bune proiecte prezentate au reușit să-și îmbunătățească poziția pe piață, iar cei mai buni producători au avut posibilitatea să-și realizeze planurile de afaceri într-un mod la care nici măcar nu au sperat înainte. Dacă vizitați **www.nonviolence21.com** aveți posibilitatea să vedeți proiectul echipei câștigătoare, care a anunțat că imediat după câștigarea premiului a primit numeroase oferte de muncă și membrii echipei au avut posibilitatea să-și îndeplinească visurile.

Ediția din acest an a competiției va începe cu *Pavilionul Top Talent*, un târg expozițional al celor 19 echipe câștigătoare de anul trecut și cu *sesiunea de nivel înalt EUROPRIX* care va avea loc *Milia*, *Cannes*, cel mai important loc al *Europei* destinat conținuturilor multimedia. În cadrul acestei sesiuni, se va discuta despre piața potențială a produselor care vor fi premiate.

În acest an va fi organizat un târg sub forma mai multor expoziții care se vor desfășura în diferite locuri. Primul loc în care se va desfășura o expoziție va fi galeria *Universității de Nave și Proiectare* din *Budapesta*, *Ungaria*. Această expoziție va avea loc între 30 aprilie și 23 mai. Celelalte expoziții vor fi anunțate ulterior și se vor desfășura pe durata întregului an.

Sesiunea de înscrieri la ediția din acest an a competiției va începe cu data de 1 aprilie. Un juriu internațional va evalua toate produsele în decurs de patru zile din luna august.

În luna noiembrie va avea loc tot la *Viena, Austria Festivalul Top Talent*.

An de an, acest festival atrage producători europeni, specialiști în consultanță, experți în comunicații și lideri în afaceri interesați de realizările tinerii generații. Festivalul reprezintă de fapt cheia viitoarelor realizări pe plan multimedia de pe continentul european.

Dl. prof. Peter A. Bruck a concluzionat că: *În nici un alt loc și în nici un alt context nu pot fi obținute rezultate mai bune pentru proiectele propuse. Viitorul tinerei scene este construit la și în jurul acestui festival.*

Cele mai bune echipe de anul acesta nu vor fi lăsate cu succesul obținut, ci dimpotrivă, ele vor fi promovate în cadrul *Pavilionului Top Talent* și târgului din anul 2005.

Sponsorii din acest an

Sponsorii sunt parteneri ai proiectului *Premiile celor mai talentați*. Până acum sponsorii anunțați sunt:

- *Adobe Systems Incorporated* – **www.adobe.de**
- *Sony DADC* – **www.sonydadc.com**
- *ATI Technologies* – **www.ati.de**

În final...

Pentru mai multe detalii și întrebări legate de competiția organizată de către *EUROPRIX* vă rugăm să contactați pe *d-na Verena Bstieler* la *ICNM* (email: **bstieler@icnm.net**, telefon: +43-662-630408-30) sau vizitați *site*-ul *web* actualizată permanent **www.toptalent.europrix.org**.

TTA 04

EUROPRIX Multimedia
Top Talent Award 2004

31 martie 2004, Praga

Finala ACM

Pe data de 31 martie a avut loc la Praga etapa finală a concursului de programare pe echipe destinat studenților ACM. La această competiție au participat 73 de echipe de studenți din toată lumea. Aceste echipe au fost selectate dintr-un număr de 3150 de echipe din 1411 universități în urma fazelor regionale ale concursului. În continuare vă prezentăm versiunea în limba română, realizată de redacța GInfo, a celor 10 probleme propuse spre rezolvare la acest concurs.

P040401: Carl Furnica

Furnicile lasă urme chimice pe pământ pentru a marca drumul pe care îl străbat pentru ca alte furnici să le urmeze. De obicei, aceste urme marchează un drum drept, dar într-o colonie de furnici există o furnică puțin neobișnuită pe nume *Carl*. *Carl* va merge destul de des în zig-zag fără nici un motiv, uneori intersecându-și traseul de mai multe ori. Când alte furnici ajung la o intersecție, ele urmează întotdeauna drumul cu cel mai puternic miros, care este și cel mai recent drum care indică furnicilor în ce direcție să părăsească intersecția respectivă.

Furnicile au lungimea de un centimetru, înaintează cu viteza de 1 cm/s și urmează cu exactitate drumurile (când ocolesc un colț, descriu unghiuri de 90 de grade). Furnicile nu se pot călca sau depăși unele pe altele.

Dacă două furnici ajung într-un punct dintr-o intersecție în același timp, atunci are prioritate de trecere cea care a parcurs cel mai lung drum făcut de *Carl*. Dacă nu există o astfel de furnică, atunci are prioritate cea care a așteptat cel mai mult timp în intersecție.

Carl iese de sub pământ în originea sistemului de coordonate la momentul de timp $t = 0$ secunde. Apoi el construiește drumul, iar la sfârșit intră sub pământ. Restul furnicilor urmează drumul la intervale regulate de timp.

Dându-se descrierea drumului lui *Carl* și momentele când restul furnicilor urmează drumul, trebuie să determinați în cât timp mulțimea de furnici va termina de parcurs drumul. Se garantează faptul că toate furnicile vor parcurge drumul în întregime.

Date de intrare

Fișierul de intrare **carl.in** conține mai multe cazuri de test. Prima linie a fișierului de intrare conține un singur număr întreg care reprezintă numărul de teste care urmează.

Datele fiecărui test sunt scrise pe mai multe linii. Astfel, prima linie a unui test conține trei numere întregi pozitive n, m și d, separate între ele printr-un spațiu, care reprezintă numărul de segmente ale drumului, numărul de furnici care urmează drumul (inclusiv *Carl*), respectiv intervalul de timp la care o furnică iese din pământ și urmează drumul.

Carl pornește la momentul de timp 0, următoarea furnică pornește după d secunde, următoarea după $2 \cdot d$ secunde ș.a.m.d. Atunci când canalul de ieșire este blocat, furnicile vor ieși din pământ cât mai repede posibil, în ordinea corectă.

Fiecare dintre următoarele n linii ale unui test conține două numere întregi x și y, separate între ele printr-un singur spațiu, care reprezintă coordonatele capătului unui segment din drumul lui *Carl*, în ordinea în care *Carl* construiește drumul.

Originea primului segment este dată de punctul de coordonate (0, 0), iar originea oricărui alt segment este dată de capătul segmentului precedent.

Pentru simplitate, *Carl* călătorește tot timpul pe segmente paralele cu axele de coordonate și nici un capăt al unui segment nu se află pe alt segment decât dacă este unul dintre capetele acestuia din urmă.

Date de ieșire

Pentru fiecare test rezultatele trebuie furnizate la *ieșirea standard*, după descrierea următoare:

```
Case C:
Carl finished the path at time t1
The ants finished in the following order:
```

```
a1 a2 a3 ... am
The last ant finished the path at time t2
```

unde `c` reprezintă numărul testului (începând cu 1); `a1`, `a2`, ..., `am` reprezintă numerele de ordine ale furnicilor în ordinea în care acestea termină de parcurs drumul și intră înapoi în pământ; `t1` și `t2` reprezintă momentele de timp la care *Carl* și ultima furnică termină de parcurs drumul.

Rezultatele furnizate pentru teste trebuie separate între ele printr-o linie vidă.

Restricții și precizări

- $1 \le n \le 50$;
- $1 \le m, d \le 100$;
- coordonatele care reprezintă capetele segmentelor drumului lui *Carl* sunt numere întregi cuprinse între -100 și 100.

Exemplu

carl.in

```
2
4 7 4
0 4
2 4
2 2
-2 2
4 7 2
0 4
2 4
2 2
-2 2
```

Ieșire standard

```
Case 1:
Carl finished the path at time 13
The ants finished in the following order:
0 2 1 3 4 5 6
The last ant finished the path at time 29

Case 2:
Carl finished the path at time 13
The ants finished in the following order:
0 4 1 5 2 6 3
The last ant finished the path at time 19
```

P040402: Heliport

În această eră a vitezei, companiile investesc în heliporturi pentru a reduce timpul călătoriilor membrilor conducerii. Heliporturile sunt de obicei zone de aterizare circulare plasate pe acoperișurile sediilor principale ale companiilor.

Sarcina voastră este să scrieți un program care determină raza cea mai mare a unui heliport circular care poate fi construit pe acoperișul unei clădiri care are forma unui poligon simplu.

Deoarece aceasta este doar faza de proiectare a construcției, programul trebuie să determine doar raza heliportului.

Date de intrare

Fișierul de intrare **heliport.in** conține mai multe teste.

Datele pentru fiecare test constau în două linii. Prima linie conține un număr întreg n care reprezintă numărul de laturi ale poligonului care descrie forma acoperișului clădirii. Cea de-a doua linie conține n perechi de forma (m, d).

Presupunând că acoperișul este desenat în plan, m este un număr întreg care reprezintă lungimea unei laturi și d este un caracter care reprezintă direcția în care se va desena această latură. Desenarea laturilor se va face în sens trigonometric pornind de la punctul în care s-a ajuns cu desenatul la pasul anterior. Caracterul d poate avea valorile U, R, D sau L care semnifică faptul că latura se va desena în sus, la dreapta, în jos sau la stânga.

Segmentele care reprezintă laturile poligonului sunt paralele cu axele de coordonate și sunt date în ordine trigonometrică. Poziția de început a desenului este dată de punctul de coordonate (0, 0).

După ultimul test din fișierul de intrare se va afla valoarea 0 pe o singură linie.

Date de ieșire

Pentru fiecare test din fișierul de intrare se va tipări la *ieșirea standard*, pe o linie numărul de ordine al acestuia (începând cu 1) și un număr real rotunjit la două zecimale exacte care reprezintă raza heliportului care poate fi construit.

Liniile furnizate la *ieșirea standard* trebuie separate între ele printr-o linie vidă și trebuie să aibă formatul:

```
Case Number # radius is: r
```

unde # reprezintă numărul testului, iar `r` este raza determinată pentru heliportul de la testul respectiv.

Restricții și precizări

- $4 \le n \le 20$;
- lungimea unei laturi a poligonului care descrie forma acoperișului clădirii este un număr întreg cuprins între 1 și 50.

Exemplu

heliport.in

```
4
2 R 2 U 2 L 2 D
10
10 R 10 U 10 L 10 U 10 R 5 U 30 L 20 D 20 R 5 D
0
```

Ieșire standard

```
Case Number 1 radius is: 1.00

Case Number 2 radius is: 10.00
```

Explicație

În figura 1 se poate observa harta acoperișului clădirii pentru cel de-al doilea set de date, iar raza maximă a heliportului este 10.

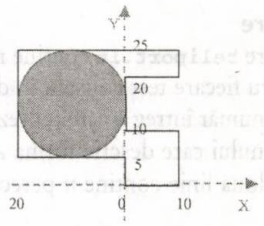


Figura 1

P040403: Imaginea este totul

Compania la care lucrați construiește roboți care pot transporta obiecte ușoare. Un astfel de robot trebuie să aibă inteligența necesară pentru a determina dacă un anumit obiect este suficient de ușor pentru a putea fi transportat. Acest lucru se realizează prin fotografierea obiectului din cele șase direcții nord, sud, est, vest, sus și jos și apoi, prin calcularea unei limite superioare a greutății obiectului în funcție de aceste imagini.

Sarcina voastră este să scrieți un program care realizează acest lucru pentru robot. Se poate presupune că fiecare obiect este format din cubulețe de dimensiune $1 \times 1 \times 1$ dispuse într-o matrice tridimensională de dimensiune $N \times N \times N$. Din această matrice tridimensională unele cubulețe pot lipsi. Fiecare cubuleț de dimensiune $1 \times 1 \times 1$ are greutatea de un gram și este desenat cu o singură culoare.

Obiectul nu este neapărat conex.

Date de intrare

Datele de intrare se citesc din fișierul **image.in** care conține mai multe teste. Pentru fiecare test se vor citi mai multe linii. Pe prima linie corespunzătoare unui test se află un număr natural N care reprezintă dimensiunile cubului.

Următoarele N linii conțin cele șase imagini de dimensiune $N \times N$ în ordinea *față*, *stânga*, *spate*, *dreapta*, *sus* și *jos*. Pe fiecare dintre aceste N linii se află șase grupuri, separate între ele prin spațiu, a câte N caractere fiecare. Un grup reprezintă o linie a uneia dintre cele șase imagini. Caracterele imaginilor pot fi litere mari ale alfabetului englezesc sau caracterul "." (punct). Literele mari descriu culoarea cubulețului respectiv, iar caracterul "." indică faptul că în zona respectivă este spațiu liber.

După ultimul test se va afla valoarea 0 pe o singură linie.

Date de ieșire

La *ieșirea standard* se vor afișa greutățile obiectelor în ordinea în care acestea au fost citite folosind formatul: `Maximum weight: # gram(s)`, unde # reprezintă greutatea maximă pe care o poate avea un obiect.

Restricție

- $1 \le N \le 10$;

Exemplu

image.in

```
3
.R. YYR .Y. RYY .Y. .R.
GRB YGR BYG RBY GYB GRB
.R. YRR .Y. RRY .R. .Y.
2
ZZ ZZ ZZ ZZ ZZ ZZ
ZZ ZZ ZZ ZZ ZZ ZZ
0
```

Ieșire standard

```
Maximum weight: 11 gram(s)
Maximum weight: 8 gram(s)
```

P040404: Nesiguranță în Praga

Praga este un oraș periculos pentru dezvoltatorii de scheme criptografice. În anul 2001 o echipă de cercetători din *Praga* au anunțat câteva lipsuri ale faimosului protocol de criptare *PGP* (*Pretty Good Privacy*). În anul 2003 au fost descoperite lipsuri ale protocoalelor *SSL* și *TLS* (*Secure Sockets Layer* și *Transport Layer Security*). Cu toate acestea, reputația *Pragăi* referitoare la protocoalele criptografice nu i-a împiedicat pe criptograful amator și încăpățânatul *Immanuel Kant-DeWitt* (cunoscut de prietenii săi ca *I. Kant-DeWitt*) să-și aducă ultima sa schemă de criptare la *Praga*.

Această schemă funcționează astfel: când se transmite un mesaj p de lungime n, transmițătorul alege un număr întreg $m \ge 2 \cdot n$ și numerele întregi s, t, i și j, unde $0 \le s, t, i, j < m$ și $i < j$. m reprezintă lungimea textului cifrat c care va fi transmis.

Inițial c conține m locuri libere. Prima literă a mesajului p va fi scrisă pe poziția s din c. A k-a ($k \ge 2$) literă din p va fi pusă în c cu i poziții libere mai departe de cea în care a fost pusă cea de-a (k-1)-a literă a lui p, numărând poziții și de la începutul lui c dacă este necesar. Pozițiile care conțin litere nu sunt considerate a fi libere.

De exemplu, dacă mesajul este PRAGUE și $s = 1$, $i = 6$ și $m = 15$ atunci literele sunt plasate în c astfel:

A	P	_	U	_	_	_	_	R	G	_	_	E	_	_
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Începând cu poziția t sau prima poziție liberă începând căutarea cu prima poziție din dreapta lui t, mesajul p mai este introdus încă o dată în c, dar de data aceasta numărând câte j poziții între litere.

De exemplu, dacă $t = 0$ și $j = 8$ a doua copie a lui p este inserată astfel (începând cu poziția 2, care este prima poziție liberă de la $t = 0$):

A	P	P	U	R	_	A	U	R	G	_	E	_	G	E
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

În final, pozițiile care rămân libere sunt ocupate cu litere alese aleator:

A	P	P	U	R	A	A	U	R	G	_	E	_	G	E	W	_	E
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Kant-DeWitt este de părere că duplicarea mesajului combinată cu utilizarea de litere aleatoare va păcăli sche-

mele de decriptare care se bazează pe frecvența de apariție a literelor și că, dacă nu se cunosc valorile *s* și *i*, nimeni nu poate afla care este mesajul original.

Sarcina voastră este să îi demonstrați lui *Kant-DeWitt* că nu are dreptate. Dându-se un număr de texte cifrate (și nici o altă informație adițională), determinați cel mai lung mesaj care se poate codifica folosind metoda lui *Kant-DeWitt*.

Date de intrare

Fișierul de intrare **`insecure.in`** conține mai multe mesaje cifrate, câte unul pe linie. Fiecare mesaj este format din litere mari ale alfabetului englezesc și nu este precedat sau succedat de spații. Fiecare mesaj va avea lungimea cuprinsă între 1 și 40 de caractere.

Ultimul mesaj cifrat va fi urmat de o linie pe care se va afla litera `X`.

Date de ieșire

La *ieșirea standard*, pentru fiecare mesaj cifrat se va afișa numărul de ordine al acestuia și cel mai lung șir de caractere care ar fi putut fi criptat sub forma prezentă în fișierul de intrare. Dacă există mai mult de un șir de caractere de lungime maximă care pot fi codificate la fel, va trebui să tipăriți fraza `"Codeword not unique"`.

Fiecare linie din fișierul de ieșire va avea formatul:

```
Code #: s
```

sau

```
Code #: Codeword not unique
```

unde `#` reprezintă numărul de ordine al mesajului cifrat în fișierul de intrare, iar `s` reprezintă mesajul original.

Exemplu

`insecure.in`

```
APPURAAURGEGEWE
ABABABAB
THEACMPROGRAMMINGCONTEST
X
```

Ieșire standard

```
Code 1: PRAGUE
Code 2: Codeword not unique
Code 3: Codeword not unique
```

P040405: Date care se suprapun

Un grup de cercetători dezvoltă un program pe calculator care va extrage date mai vechi, legate de cotațiile bursiere de pe piață, folosind un serviciu care cere o taxă fixă pentru fiecare cotație bursieră pe care o furnizează.

Grupul a analizat colecția de cotații extrase anterior și a descoperit multe duplicate, de unde rezultă că au fost irosiți bani.

Așadar, noul program va trebui să mențină o listă a cotațiilor cerute anterior de membrii grupului iar în momentul în care sunt necesare alte cotații, va extrage doar cotațiile din datele pentru care nu există informații, minimizând astfel costul.

Sarcina voastră este să scrieți un program care determină când trebuie extrase cotații noi. Datele de intrare pentru program vor consta în intervale de timp pentru care, în trecut, au fost cerute cotații bursiere și intervale de timp pentru care sunt necesare noi cotații. Programul va determina intervalele de timp pentru care trebuie extrase cotații folosind serviciul cu taxă.

Date de intrare

Fișierul de intrare **`intersect.in`** va conține mai multe seturi de date.

Fiecare set de date este descris pe mai multe linii. Prima linie conține două numere întregi *NX* și *NR*, separate între ele printr-un singur spațiu, care reprezintă numărul de intervale de timp pentru care există informații, respectiv numărul de intervale de timp pentru care sunt necesare informații. Următoarele *NX* + *NR* linii conțin perechi de date. Prima dată dintr-o pereche este mai mică sau egală cu cea de-a doua. Primele *NX* perechi de date reprezintă intervalele de timp pentru care există informații, iar restul de *NR* perechi reprezintă intervalele de timp pentru care sunt necesare informații legate de cotațiile bursiere. Cele două date dintr-o pereche sunt separate între ele printr-un singur spațiu.

Ultimul set de date din fișierul de intrare este urmat de o linie pe care se află două valori `0` separate între ele printr-un singur spațiu.

Fiecare dată din fișierul de intrare va avea formatul *YYYYMMDD*, unde *YYYY* reprezintă anul (care este cuprins între 1700 și 2100), *MM* reprezintă luna (care poate avea valorile 01, 02, ..., 12), iar *DD* reprezintă ziua (care reprezintă o zi validă în raport cu anul și luna). Ziua poate avea valoarea minimă 01, valoarea maximă 31 pentru lunile 01, 03, 05, 07, 08, 10 și 12, valoarea maximă 30 pentru lunile 04, 06, 09 și 11, valoarea maximă 28 pentru luna 02, dacă anul nu este bisect, și valoarea maximă 29 pentru luna 02, dacă anul este bisect. Un an este bisect dacă este divizibil cu 4 și nu este divizibil cu 100 sau dacă este divizibil cu 400.

Date de ieșire

La ieșirea standard, pentru fiecare set de date din fișierul de intrare se va scrie numărul de ordine al acestuia pe o linie urmată de mai multe linii pe care se vor scrie intervalele de timp pentru care trebuie extrase informații sau o linie pe care se va afla textul `No additional quotes are required`, ca în exemplu. Rezultatele furnizate pentru fiecare set de date vor fi separate între ele printr-o linie vidă.

Restricții

- $0 \le NX, NR \le 100$.

Exemplu

`intersect.in`

```
1 1
19900101 19901231
```

```
19901201 20000131
0 3
19720101 19720131
19720201 19720228
19720301 19720301
1 1
20010101 20011231
20010515 20010901
0 0
```

Ieșire standard

```
Case 1:
    1/1/1991 to 1/31/2000
Case 2:
    1/1/1972 to 2/28/1972 .
    3/1/1972
Case 3:
    No additional quotes are required.
```

P040406: Alipind hărți

Fotografiile unei zone, luate din avion sau satelit, pentru care trebuie realizată o hartă, au de obicei o rezoluție mare pentru a identifica în mod unic caracteristicile majore pe care aceasta la deține. Din moment ce o fotografie acoperă doar o mică porțiune a pământului, realizarea unei hărți pentru o zonă mai mare necesită mai multe fotografii care se suprapun pe porțiuni și prin alipirea fotografilor rezultă o hartă a unei zone mai mari.

Pentru această problemă aveți la dispoziție câteva hărți dreptunghiulare, fiecare fiind reprezentat sub forma unui tablou bidimensional de caractere. O celulă poate conține o literă majusculă a alfabetului englezesc în cazul în care corespunde unei zone care conține o caracteristică care identifică în mod unic zona. Litere diferite corespund unor caracteristici diferite, dar aceeași caracteristică poate identifica mai multe celule (de exemplu, o șosea). O celulă conține caracterul '-' dacă în locul respectiv nu există o caracteristică importantă. Alipirea a două hărți constă în suprapunerea lor astfel încât una sau mai multe caracteristici importante să fie suprapuse. O celulă care conține o caracteristică importantă pe una din hărți, poate fi suprapusă peste o celulă, de pe altă hartă, care nu conține o caracteristică importantă, dar nu pot fi suprapuse două celule care conțin caracteristici importante distincte.

```
          --A-C  C----  C----  ----D  -D--C
          ----D  D---F  -----  -E--B  ----G
          ----B  B----  B-A-C  -----  ----B
Harta nr.:  1      2      3      4      5
```

Dacă se consideră hărțile de mai sus care au trei linii și cinci coloane fiecare, se poate observa că cea mai din dreapta coloană a primei hărți se potrivește perfect cu prima coloană a celei de-a doua hărți. Așadar, prima și a doua hartă pot fi suprapuse și rezultă o hartă care are trei linii și nouă coloane. Prima hartă se mai poate suprapune și cu harta a treia din moment ce caracteristicile importante `B` și `C` de pe ultima coloană a primei hărți se potrivesc cu cele de pe prima coloană a celei de-a treia hărți; caracteristica `D` nu se potrivește perfect peste caracterul '-', dar nu există nici un conflict. Similar, prima linie a hărții 1 se poate suprapune cu ultima linie a hărții 3.

"Scorul" unei perechi de hărți indică suprafața pe care se potrivesc cele două hărți. Scorul suprafeței suprapuse a două hărți reprezintă numărul de celule care conțin caracteristici importante, care coincid în suprapunere și care dau cea mai bună potrivire. Scorul unei perechi de hărți este dat de scorul maxim care se poate obține pentru o suprapunere validă. Așadar, scorul unei perechi de hărți care au fiecare trei linii și cinci coloane este un număr întreg cuprins între `0` și `15`.

Un "deplasament" reprezintă o pereche de numere întregi (r, c) care indică modul în care două hărți a și b se pot suprapune. Valoarea r reprezintă deplasamentul liniilor din b relativ la liniile din a, iar valoarea c reprezintă deplasamentul coloanelor din b relativ la coloanele din a.

De exemplu, suprapunerea hărților 1 și 2 are deplasamentul `(0, 4)` și scorul 3, iar cele două suprapuneri posibile ale hărților 1 și 3 cu scorul 2 au deplasamentele `(0, 4)`, respectiv `(-2, 0)`.

Următorii pași descriu modul în care se realizează alipirea unui secvențe de hărți:

- se suprapun hărțile care au cel mai mare scor pozitiv (se rezolvă conflictele folosind harta cu cel mai mic număr de ordine);
- se elimină hărțile care au fost suprapuse din secvență;
- se adaugă rezultatul suprapunerii celor două hărți în secvență, atribuindu-i-se cel mai mare număr de ordine care apare în secvență și la care se adaugă 1.

În exemplul anterior, hărțile 1 și 2 prin suprapunere formează harta cu numărul de ordine 6, iar hărțile 1 și 2 sunt eliminate din secvență. Cei trei pași descriși anterior se repetă până când secvența conține o singură hartă sau până când nu mai pot fi suprapuse hărți (scorul oricărei perechi de hărți din secvență este `0`).

Dacă două hărți se pot suprapune în mai multe feluri cu același scor, atunci ele trebuie suprapuse în funcție de cel mai mic deplasament pentru linii și apoi în funcție de cel mai mic deplasament pentru coloane.

Date de intrare

Fișierul de intrare **`maps.in`** conține mai multe seturi de date fiecare având între două și zece hărți.

Fiecare set de date constă în mai multe linii. Prima linie a unui set de date conține un singur număr întreg care reprezintă numărul de hărți pentru setul de date respectiv. În continuare urmează hărțile.

Fiecare hartă constă în mai multe linii. Prima linie a unei hărți conține două numere întregi NR și NC, separate între

ele printr-un singur spațiu, care reprezintă numărul de linii, respectiv numărul de coloane ale hărții respective. Următoarele *NR* linii ale descrierii unei hărți conțin câte o linie a hărții care este dată doar de primele *NC* caractere de pe ea, restul caracterelor suplimentare trebuind ignorate.

Ultimul set de date este urmat de o linie pe care se va afla valoarea 0.

Date de ieșire

La ieșirea standard, pentru fiecare set de date trebuie afișat numărul său de ordine (începând cu 1) și hărțile obținute, fiecare identificată prin numărul său de ordine și trebuie mărginite de o frontieră. Furnizarea rezultatelor trebuie să se facă folosind formatul din exemplu. Nici o hartă nu trebuie să aibă mai mult de 70 de coloane.

Restricții

- $1 \le NR, NC \le 10$.

Exemplu

maps.in

```
5
3 5
--A-C
----D
----B
3 5
C----
D---F
B----
3 5
C----
-----
B-A-C
3 5
----D
-E--B
-----
3 5
-D--C
----G
----B
2
3 5
----A
----B
----C
3 5
A----
B----
D----
0
```

Ieșire standard

```
Case 1
    MAP 9:
    +--------------+
    |-D--C---------|
    |----G---------|
    |----B-A-C-----|
    |----B-A-C-----|
    |-----E--B-----|
    |--------------|
    +--------------+
Case 2
    MAP 1:
    +------+
    |----A|
    |----B|
    |----C|
    +------+
    MAP 2:
    +------+
    |A-----|
    |B-----|
    |D-----|
    +------+
```

P040407: Navigare

GPS-ul este un sistem de navigare care se bazează pe un set de sateliți care se află pe orbite la aproximativ 20.000 de km deasupra pământului. Fiecare satelit descrie o orbită cunoscută și transmite semnale radio care codifică timpul curent. Dacă un vehicul echipat cu un dispozitiv *GPS* are un ceas precis, poate compara timpul său local cu cel codificat de semnalele recepționate de la sateliți. Deoarece semnalele radio sunt transmise cu o frecvență cunoscută, vehiculul poate determina distanța dintre poziția sa și poziția satelitului în momentul în care a transmis semnalul radio. Măsurând distanța sa față de mai mulți sateliți de pe orbite cunoscute, un vehicul poate să-și determine poziția cu o precizie foarte mare.

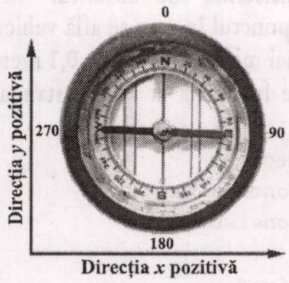


Figura 2

Sarcina voastră este să scrieți un program simplu de "pilot-automat" care se bazează pe navigare *GPS*. Pentru ca problema să fie mai simplă, vom presupune că totul se desfășoară în plan. Cu alte cuvinte, nu trebuie luată în considerare curbura pământului sau altitudinea sateliților. Mai mult, în problemă se folosesc viteze care sunt mai apropiate celor ale avioanelor și undelor sonore decât cele ale sateliților și undelor radio.

Dându-se un set de semnale ale unor surse mobile, programul vostru trebuie să calculeze poziția în care se recepționează semnalul în plan. Apoi, dându-se un punct de destinație în plan, programul trebuie să determine direcția indicată de o busolă și care trebuie urmată pentru a ajunge din punctul determinat până în punctul destinație. Toate direcțiile busolei sunt în grade. Gradul 0 corespunde direcției nordului, gradul 90 corespunde direcției est, gradul 180 corespunde direcției sud și gradul 270 corespunde direcției vest, după cum se poate observa și în figura 2.

Date de intrare

Fișierul de intrare **navigation.in** conține mai multe seturi de date.

Prima linie a fiecărui set de date conține un număr întreg pozitiv *N*, care reprezintă numărul de surse care emit semnale, urmat de trei numere reale *t*, *x* și *y* care reprezintă momentul de timp local exprimat în secunde când semnalele au fost primite relativ la momentul de timp 0, iar *x* și *y* reprezintă coordonatele punctului destinație. Numerele de pe această linie sunt separate între ele printr-un spațiu.

Fiecare dintre următoarele *N* linii conține patru numere reale, separate între ele printr-un spațiu, care conțin informațiile legate de o sursă. Primele două numere reprezintă poziția sursei la momentul de timp 0. Al treilea număr re-

prezintă direcția de deplasare D a sursei exprimată în grade, iar al patrulea număr reprezintă timpul codificat de sursă.

Ultimul set de date este urmat de o linie pe care se află patru valori 0 separate între ele printr-un spațiu.

Date de ieșire

La ieșirea standard, pentru fiecare set de date se va afișa pe o linie numărul de ordine al setului din fișierul de intrare și direcția în grade care trebuie urmată pentru a se ajunge la destinație, cuvântul "`Inconclusive`", dacă direcția nu poate fi determinată, cuvântul "`Inconsistent`", dacă datele sunt inconsistente sau cuvântul "`Arrived`", dacă distanța dintre punctul în care se află vehiculul și punctul de sosire este mai mică sau egală cu 0,1 metri.

Fiecare linie furnizată la ieșirea standard poate avea forma:

```
Trial #: D degrees
Trial #: Inconclusive
Trial #: Inconsistent
```

sau

```
Trial #: Arrived
```

unde # reprezintă numărul setului de date, iar D reprezintă direcția de deplasare exprimată în grade.

Restricții și precizări

- $1 \le N \le 10$;
- numerele din fișierul de intrare nu sunt mai mari decât 10.000 și nici un număr real nu are mai mult de 5 zecimale;
- toate numerele care reprezintă coordonate sunt date în metri;
- sursele care emit semnal se deplasează cu 100 de metri pe secundă, iar semnalele se propagă cu 350 de metri pe secundă;
- datorită pierderii preciziei în momentul sincronizării ceasurilor, distanțele se pot calcula cu o eroare de 0,1 metri, mai exact, dacă distanța dintre două puncte este mai mică sau egală cu 0,1 metri se poate considera că acestea sunt identice;
- există posibilitatea ca un semnal să fie corupt la transmitere, ceea ce conduce la faptul că datele mai multor semnale să fie inconsistente.

Exemplu

navigation.in

```
3 2.53571 1050.0 1050.0
-100.0 350.0 90.0 1.75
350.0 -100.0 0.0 1.75
350.0 800.0 180.0 1.75
2 2.0 1050.0 1050.0
-100.0 350.0 90.0 1.0
350.0 -100.0 0.0 1.0
0 0 0 0
```

Ieșire standard

```
Trial 1: 45 degrees
Trial 2: Inconclusive
```

Explicație

Figura 3 corespunde primului set de date din fișierul de intrare. Pozițiile celor 3 sateliți la momentul de timp $t = 0$ sunt $A(-100, 350)$, $B(350, -150)$ și $C(350, 800)$. Semnalele recepționate de dispozitivul *GPS* au fost transmise la momentul $t = 1{,}75$ când sateliții erau în pozițiile A', B', respectiv C' (în general, semnalele nu sunt transmise în același timp de către toți sateliții). Semnalele de la cei trei sateliți converg către punctul D la momentul de timp $t = 2{,}53571$ ceea ce însemnă că D reprezintă poziția vehiculului. Din punctul D trebuie urmată direcția de 45 de grade indicată de busolă pentru a ajunge în punctul destinație de coordonate (1050, 1050).

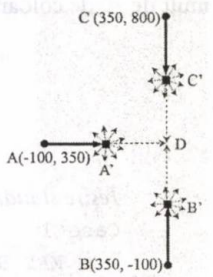


Figura 3

P040408: Copacii de pe străzi

Consiliul local al orașului *Greenville* a votat recent îmbunătățirea imaginii străzilor din oraș. Pentru a oferi mai multă verdeață peisajului, consiliul a decis să planteze copaci de-a lungul celor mai importante străzi și bulevarde. Pentru a putea aproxima costul acestei îmbunătățiri urbane, consiliul dorește să știe câți copaci vor fi plantați. Plantarea copacilor trebuie să respecte următoarele restricții:

- de-a lungul unei străzi copacii trebuie plantați la o distanță de minim 50 de metri unii de alții pentru a li se oferi o creștere și dezvoltare normală și pentru a menține costul acestei îmbunătățiri între niște limite rezonabile;
- din motive de siguranță nu trebuie plantați copaci la mai puțin 25 de metri de cea mai apropiată intersecție pentru ca participanții la trafic să se poată vedea unii pe alții cu ușurință când se apropie de intersecție. Siguranța traficului nu trebuie să fie compromisă de reducerea vizibilității.

Toate străzile considerate sunt drepte. Nu există curbe de-a lungul lor.

Consiliul orașului dorește să cunoască numărul maxim de copaci care pot fi plantați în conformitate cu cele două restricții.

Date de intrare

Fișierul de intrare **streets.in** conține mai multe hărți de străzi.

Prima linie a unei hărți conține un singur număr întreg n care reprezintă numărul de străzi ale hărții. Fiecare dintre următoarele n linii descrie o stradă ca un segment de dreaptă din plan. O linie a fișierului de intrare care descrie

o stradă conține patru numere întregi x_1, y_1, x_2 și y_2, separate între ele printr-un spațiu, cu semnificația că strada curentă merge din punctul de coordonate (x_1, y_1) până în punctul de coordonate (x_2, y_2).

Ultima descriere a unei hărți de străzi este urmată de o linie care conține valoarea `0`.

Date de ieșire

La ieșirea standard trebuie furnizate, pentru fiecare hartă, numărul de ordine al hărții din fișierul de intrare și numărul maxim de copaci care pot fi plantați astfel încât să fie respectate restricțiile din enunț.

Pentru fiecare hartă se vor scrie două linii. Prima linie trebuie să aibă formatul:

```
Map #
```

unde `#` reprezintă numărul de ordine al hărții din fișierul de intrare, începând cu 1 ca în exemplu, iar cea de-a doua linie trebuie să aibă formatul:

```
Trees = c
```

unde `c` reprezintă numărul maxim de copaci care pot fi plantați pe străzile care aparțin unei hărți.

Restricții și precizări

- $1 \le n \le 100$;
- coordonatele punctelor care descriu capetele unei străzi sunt numere întregi cuprinse între 0 și 1.000.000;
- toate străzile au lungimi pozitive și fiecare capăt aparține unei singure străzi;
- pentru fiecare stradă distanțele dintre două intersecții vecine nu sunt multipli exacți de 25 de metri; mai exact, diferența dintre o astfel de distanță și cel mai apropiat multiplu de 25 va fi cel puțin 0.001 metri.
- în fiecare intersecție se întâlnesc exact două străzi.

Exemplu

streets.in

```
3
0 40 200 40
40 0 40 200
0 200 200 0
4
0 30 230 30
0 200 230 200
30 0 30 230
200 0 200 230
3
0 1 121 1
0 0 121 4
0 4 121 0
0
```

Ieșire standard

```
Map 1
Trees = 13
Map 2
Trees = 20
Map 3
Trees = 7
```

P040409: Suspans!

Jan și *Tereza* locuiesc în clădiri vecine și apartamentele lor sunt față în față. Pentru proiectul de la disciplina științe de la școală ei doresc să construiască un pod suspendat din cablu, sfoară și carton care să facă legătura între cele două clădiri. Două bucăți identice ca lungime formează cablurile principale de suspensie și sunt atașate de marginea de jos a ferestrelor lor. Platforma de carton care reprezintă suprafața netedă a podului este susținută de mai multe sfori care sunt legate de cablurile principale de suspensie. Podul orizontal se află la exact un metru sub cel mai de jos punct al cablurilor de suspensie. Din motive estetice platforma podului ar trebui să fie la cel puțin doi metri sub cea mai de jos muchie a celei mai de jos ferestre a fiecăruia dintre cei doi elevi. Legile fizicii spun că fiecare cablu de suspensie are forma unei parabole.

Cu toate că *Jan* și *Tereza* nu au de gând să se plimbe pe acest prototip de pod, apare o problemă serioasă: unii locatari ai celor două imobile au pisici, iar alții au canari. Cei doi doresc să fie siguri că podul lor nu va putea permite unei pisici să ajungă la un canar. Cei doi au observat că o pisică nu poate sări pe ceva ce este mai înalt de 0,5 metri și nu poate sări de la o înălțime mai mare de trei metri. Deci, atâta timp cât podul se află la mai mult de 0,5 metri deasupra marginii de jos a geamului la care se află o pisică sau la cel puțin trei metri sub marginiea de jos a geamului la care se află o pisică, atunci aceasta nu va sări. De asemenea, dacă o pisică reușește să sară pe pod, atunci nu va putea ajunge la geamul unui canar dacă podul se află la mai mult de 0,5 metri sub marginea de jos a geamului la care se află canarul sau la cel puțin trei metri deasupra marginii de jos a geamului la care se află canarul. Pisicile sunt interesate doar să ajungă la canari și nu să se întoarcă acasă.

Figura de mai jos arată că apartamentul lui *Jan*, etichetat cu *J* și cel al *Terezei*, etichetat cu *T*, sunt legate printr-un cablu care unește punctele cele mai de jos ale ferestrelor lor și podul propriu-zis se află la un metru dedesubtul celui mai de jos punct al cablului de suspensie. Pisica de la etajul al doilea poate ajunge la canarul de la etajul al doilea din clădirea vecină folosind podul.

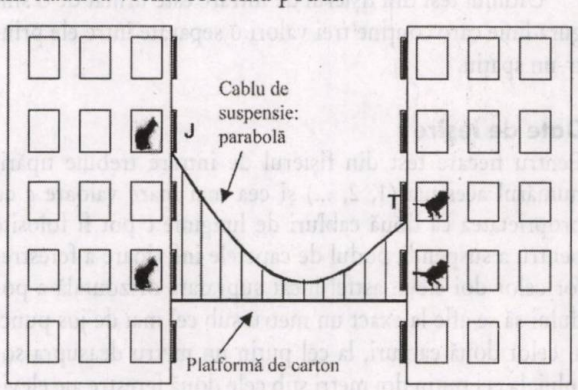


Figura 4

Sarcina voastră este să scrieți un program care determină de câtă sfoară au nevoie cei doi pentru a realiza fiecare segment de cablu al podului astfel încât să nu pună în pericol nici un canar din cele două clădiri.

Programul trebuie să țină cont de:

- distanța, exprimată în metri, dintre cele două imobile;
- numerele etajelor la care locuiesc cei doi elevi;
- tipul de animale care se află la etajele dedesubtul etajului la care stă *Jan*;
- tipul de animale care se află la etajele dedesubtul etajului la care stă *Tereza*.

Programul trebuie să determine lungimea celui mai lung cablu care poate fi folosit la suspendarea podului care leagă cele două clădiri și care nu permite nici unei pisici să ajungă la un canar prin intermediul podului.

Podul orizontal trebuie să se afle la cel puțin un metru deasupra solului și la exact un metru sub cel mai de jos punct al cablului de suspensie.

De asemenea, podul trebuie să se afle la cel puțin doi metri sub cea mai de jos fereastră a celor doi.

Date de intrare

Fișierul de intrare **`suspense.in`** constă în mai multe teste. Fiecare test este descris prin trei linii.

Prima linie pentru un test va conține două numere întregi pozitive j și t și un număr real d, separate prin spațiu, care reprezintă etajul la care locuiește *Jan*, etajul la care locuiește *Tereza*, respectiv distanța dintre cele două clădiri exprimată în metri.

Cea de-a doua linie conține j litere mari ale alfabetului englezesc separate printr-un spațiu, iar cea de-a treia linie corespunzătoare unui test conține t litere mari ale alfabetului englez separate printr-un singur spațiu. Aceste litere reprezintă tipul de vietăți care trăiesc în etajele celor două clădiri, care sunt mai mici sau egale cu etajele la care locuiesc cei doi, și pot avea valorile `B` (la etajul respectiv se află un canar), `C` (la etajul respectiv se află o pisică) sau `N` (la etajul respectiv nu se află nici una dintre cele două tipuri de vietăți de mai sus). Vietățile din cele două clădiri sunt date pe linia a doua și a treia în ordinea crescătoare a etajelor de la 1 la j, respectiv de la 1 la t.

Ultimul test din fișierul de intrare este urmat de o singură linie care conține trei valori `0` separate între ele printr-un spațiu.

Date de ieșire

Pentru fiecare test din fișierul de intrare trebuie tipărit numărul acestuia (1, 2, ...) și cea mai mare valoare c cu proprietatea că două cabluri de lungime c pot fi folosite pentru a suspenda podul de capetele inferioare a ferestrelor celor doi elevi, astfel încât suprafața orizontală a podului să se afle la exact un metru sub cel mai de jos punct al celor două cabluri, la cel puțin un metru deasupra solului, la cel puțin doi metri sub cele două ferestre ale elevilor și nu permite nici unei pisici să ajungă la un canar.

Valoarea c trebuie rotunjită la trei zecimale exacte.

Dacă nu poate fi construit un pod care să respecte condiția din enunț trebuie afișat cuvântul `impossible`.

Pentru fiecare test din enunț se va tipări o linie de forma:

```
Case #: c
```

sau

```
Case #: impossible
```

unde # reprezintă numărul testului, iar c reprezintă valoarea calculată rotunjită la trei zecimale exacte.

Liniile furnizate la ieșirea standard trebuie despărțite printr-o linie vidă, ca în exemplu.

Restricții și precizări

- $2 \le j, t \le 25$;
- $1 \le d \le 25$;
- etajele sunt identificate prin numere întregi, iar cel mai de jos etaj este identificat prin 1;
- toate camerele din cele două clădiri au exact trei metri înălțime;
- toate ferestrele ale celor două clădiri au exact 1,5 metri înălțime și capătul de jos al fiecărei ferestre se află la exact un metru deasupra podelei fiecărei camere;

Exemplu

`suspense.in`	*Ieșirea standard*

`4 3 5.0`
`N C N C`
`N B B`
`4 3 5.0`
`C B C C`
`B C B`
`0 0 0`
	`Case 1: 14.377`
	
	`Case 2: impossible`

P040410: Controlul traficului aerian

Pentru a evita coliziunile din aer, majoritatea zborurilor comerciale sunt monitorizate de la sol de centre de control a traficului aerian care înregistrează pozițiile avioanelor folosindu-se de sisteme radar.

Datele pentru această problemă constau în două mulțimi: o mulțime de avioane și o mulțime de centre de control și trebuie să determinați cum trebuie distribuită monitorizarea avioanelor între centrele de control. Poziția fiecărui avion este descrisă de o pereche unică (x, y) de coordonate. Pentru scopul acestei probleme altitudinea la care se află avioanele poate fi ignorată.

Numărul de avioane care pot fi monitorizate de către un centru de control variază din când în când datorită schimbărilor de echipament și a angajaților.

La orice moment de timp, fiecare centru de control monitorizează atâtea avioane câte poate, în funcție de următoarele priorități:

- se preferă monitorizarea avioanelor care sunt mai apropiate de centrul de control;
- dacă două avioane sunt egal depărtate de centrul de control, și centrul poate să monitorizeze numai unul dintre

ele, atunci va fi monitorizat cel mai dinspre nord (valoare pozitivă pentru coordonata *y*);
- dacă două avioane sunt egal depărtate de centrul de control și au aceeași valoare pentru coordonata *y*, atunci este prioritar cel mai dinspre est (valoare pozitivă pentru coodonata *x*).

În orice moment, fiecare centru de control dispune de o arie circulară de control a cărei rază este dată de distanța dintre el și cel mai îndepărtat avion monitorizat. Toate avioanele cuprinse în această arie de control sunt monitorizate de centrul de control respectiv. Avioanele care se află chiar pe frontiera unui centru de control pot sau nu pot fi monitorizate de acesta, în funcție de prioritățile amintite mai sus.

Voi nu veți cunoaște pozițiile centrelor de control, în schimb, pentru fiecare centru de control veți cunoaște numărul de avioane pe care acesta îl monitorizează la un moment dat și coordonatele a două puncte care se află pe frontiera ariei de control. Folosind aceste informații, veți putea calcula poziția centrului de control și puteți decide care avioane sunt monitorizate de acesta.

Dacă datele sunt consistente în raport cu două sau mai multe arii de control, trebuie aleasă aria de control care include avionul cel mai nordic, iar dacă există două avioane având aceeași coordonată *y*, se va alege aria de control care monitorizează pe cel mai estic dintre avioane.

Figura de mai jos ilustrează patru avioane și două centre de control. Fiecare centru de control este reprezentat prin două puncte de pe frontieră, etichetate cu *A* și *B*, iar avioanele sunt etichetate cu *P*1, *P*2, *P*3 și *P*4. În acest exemplu avioanele *P*1 și *P*4 sunt monitorizate de un singur centru de control, avionul *P*3 este monitorizat de două centre de control, iar avionul *P*2 nu este monitorizat de nici un centru de control.

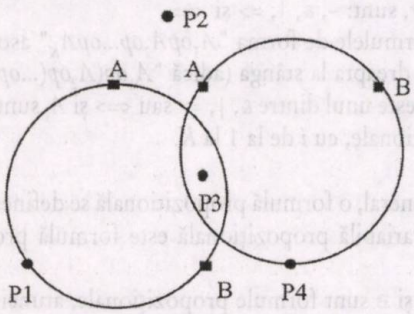


Figura 5

Date de intrare

Datele de intrare ce citesc din fișierul **traffic.in** și constau în mai multe teste. Datele pentru un test constau în mai multe linii.

Prima linie a unui anumit test conține două numere întregi *NP* și *NC*, separate printr-un singur spațiu, care reprezintă numărul de avioane, respectiv numărul centrelor de control. Fiecare dintre următoarele *NP* linii conține două numere reale *x* și *y* care reprezintă coordonatele la care se află un avion. Fiecare dintre următoarele *NC* linii conține cinci numere separate între ele prin spațiu. Primul număr este întreg, este cuprins între 0 și *NP* inclusiv și reprezintă numărul de avioane monitorizate de un centru de control; următoarele două numere sunt reale și reprezintă coordonatele primului punct aflat pe frontiera ariei de control a centrului respectiv, iar ultimele două numere sunt reale și reprezintă coordonatele celui de-al doilea punct aflat pe frontiera ariei de control a centrului.

Ultimul test este urmat de o linie pe care se află două valori 0 separate între ele printr-un singur spațiu.

Date de ieșire

Rezultatele vor fi furnizate la *ieșirea standard*.

Pentru fiecare test trebuie calculate numărul de avioane care nu sunt monitorizate de nici un centru de control, numărul de avioane care sunt monitorizate de un centru de control și așa mai departe până la numărul de avioane care sunt monitorizate de *NC* centre de control. Aceste valori vor trebui scrise pe o singură linie sub forma:

```
Trial #: n0 n1 n2 ... nNC
```

unde # reprezintă numărul testului, iar n_0, n_1, n_2, ..., n_{NC} reprezintă valorile calculate.

Dacă datele pentru un centru de control din cadrul unui anumit test sunt inconsistente, pentru testul respectiv se va scrie la ieșirea standard o linie de forma:

```
Trial #: Impossible
```

unde # reprezintă numărul testului.

Liniile furnizate la ieșirea standard trebuie despărțite printr-o linie vidă, ca în exemplu.

Restricții și precizări

- $0 < NP < 100$;
- $0 < NC < 10$;
- dacă diferența a două distanțe este mai mică decât 0,00001, atunci distanțele trebuie tratate ca fiind egale.

Exemplu

traffic.in

```
4 2
3.0 0.0
0.0 0.0
1.6 2.8
2.0 1.0
2 1.0 2.0 2.0 0.0
2 2.0 2.0 4.0 2.0
2 1
0.0 0.5
0.0 -0.5
0 -1.0 0.0 1.0 0.0
0 0
```

Ieșirea standard

```
Trial 1: 1 2 1

Trial 2: Impossible
```

Buzău, 2-9 aprilie 2004

Olimpiada NAȚIONALĂ de informatică 2004

Vă prezentăm în continuare enunțurile celor 18 de probleme propuse spre rezolvare la ediția din acest an a Olimpiadei Naționale de Informatică pentru liceeni, urmând ca în numărul următor să publicăm și problemele propuse spre rezolvare la gimnaziu și pentru lărgirea lotului național de informatică.

Clasa a IX-a

P040411: Coduri

Un detectiv particular are de rezolvat un caz special. Este vorba de o deturnare de fonduri. Pentru a putea rezolva cazul trebuie să găsească un șir cu n coduri distincte. Fiecare cod este un număr natural scris în baza 10. Din păcate lucrurile nu sunt simple, pentru că din cercetările efectuate a obținut două informații. Prima informație este legată de faptul că suma pătratelor codurilor este un cub perfect, iar a doua spune că suma cuburilor codurilor este un pătrat perfect.

Ajutați detectivul să găsească un șir de coduri $x_1, x_2, \ldots, x_n$, care verifică condițiile din enunț.

Date de intrare

Fișierul de intrare **coduri.in** conține pe prima linie numărul natural n.

Date de ieșire

Fișierul de ieșire **coduri.out** va conține n linii, câte una pentru fiecare cod din șir, în ordine crescătoare.

Restricții

- $1 \le n \le 20$;
- $x_i \le n^{14}$, $\forall\ 1 \le i \le n$.

Exemplu

coduri.in	coduri.out
2	625 1250

Timp de execuție: 1 secundă/test

P040412: Logic

Demonstrarea automată a teoremelor și verificarea satisfiabilității unei formule constituie două capitole importante în cadrul logicii matematice. Formulele propoziționale sunt alcătuite din variabile propoziționale (variabile care pot lua doar două valori: *adevărat* sau *fals*) și din operatorii logici *și*, *sau*, *negație*, *echivalent*, *implică*.

Iată câteva exemple de formule propoziționale:

```
~p&(q<=>p)=>q
p|q<=>~p&~q
p
p=>q=>a=>t=>~p
```

În acest exemplu, `p`, `q`, `t` și `a` sunt variabilele propoziționale, `~` este operatorul unar *negație*, `&` este operatorul binar *și*, `|` este operatorul binar *sau*, `=>` este *implicația* logică (apare numai în acest sens, nu apare și `<=`), iar `<=>` este *echivalența* logică. În plus, într-o formulă propozițională pot să apară și paranteze care stabilesc ordinea operațiilor. În lipsa parantezelor, operatorii, în ordinea priorității lor, sunt: `~`, `&`, `|`, `=>` și `<=>`.

În formulele de forma "$A_1 op A_2 op ... op A_K$" asocierile se fac de la dreapta la stânga (adică "$A_1 op (A_2 op (... op A_K) ...)$"), unde *op* este unul dintre `&`, `|`, `=>` sau `<=>` și A_i sunt formule propoziționale, cu i de la 1 la K.

În general, o formulă propozițională se definește astfel:

- orice variabilă propozițională este formulă propozițională;
- dacă `A` și `B` sunt formule propoziționale, atunci `(A)`, `~A`, `A&B`, `A|B`, `A=>B` și `A<=>B` sunt formule propoziționale.

Dacă înlocuim într-o formulă propozițională toate variabilele cu valori de adevăr (*adevărat* sau *fals*), obținem o afirmație. Valoarea de adevăr a unei afirmații este dată de următoarea definiție:

- dacă afirmația constă dintr-o singură valoare de adevăr, afirmația ia valoarea respectivă;
- dacă `A` și `B` sunt afirmații, atunci:
  - `A` este adevărată dacă și numai dacă valoarea sa de adevăr este *adevărat*;

- `(A)` este adevărată dacă și numai dacă `A` este adevărată;
- `~A` este falsă dacă și numai dacă `A` este adevărată;
- `A&B` este adevărată dacă și numai dacă atât `A`, cât și `B` sunt adevărate;
- `A|B` este falsă dacă și numai dacă `A` este fals și `B` este fals;
- `A=>B` este adevărată dacă și numai dacă `~A|B` este adevărată;
- `A<=>B` este adevărată dacă și numai dacă `(A=>B)&(B=>A)` este adevărată.

Se numește soluție a formulei propoziționale P (formulă în care apar numai variabilele propoziționale distincte $A_1, A_2, ..., A_N$) orice N-uplu $(v_1, v_2, ..., v_N)$ (cu v_i valori de adevăr) pentru care înlocuind fiecare variabilă A_i cu valoarea v_i, afirmația rezultată este adevărată.

Logica fiind un obiect nesuferit de studenții de la informatică, ei apelează la informaticienii din clasa a IX-a pentru a-i ajuta să numere câte soluții distincte are o formulă propozițională dată.

Date de intrare

În fișierul de intrare **`logic.in`** se găsește pe prima linie o formulă propozițională în care variabilele propoziționale sunt reprezentate de litere mici ale alfabetului englez.

Date de ieșire

În fișierul de ieșire **`logic.out`** se va scrie pe prima linie numărul de soluții pentru formula propozițională din fișierul de intrare, urmat de caracterul sfârșit de linie.

Restricții și precizări

- la intrare se va da întotdeauna o formulă propozițională corectă sintactic;
- formula are mai puțin de 232 de caractere;
- în formulă nu apar mai mult de 10 litere mici ale alfabetului latin;
- se recomandă ca afișarea soluției să se facă în modul următor:
  - *Pascal*: `writeln(f, nsol);`
  - *C*: `fprintf(f, "%d\n", nsol);`
  - *C++*: `f << nsol << "\n";`

Exemple

`logic.in`
```
p|q<=>~p&~q
```

`logic.out`
```
4
```

`logic.in`
```
a
```

`logic.out`
```
1
```

Timp de execuție: 1 secundă/test

P040413: Poligon

Se consideră un caroiaj cu dimensiunea $M \times N$ în care sunt plasate K puncte. Fiecare punct poate fi legat de vecinul său direct pe cel mult opt direcții (*N*, *NE*, *E*, *SE*, *S*, *SV*, *V*, *NV*).

Determinați patrulaterele având vârfurile în punctele date, iar laturile formate din legături între două sau mai multe puncte coliniare.

Date de intrare

Pe prima linie a fișierului de intrare **`poligon.in`** se află trei numere întregi M, N și K, separate prin spațiu, care reprezintă dimensiunile caroiajului și numărul de puncte.

Fiecare dintre următoarele K linii conține câte trei numere naturale nenule I, J și V, separate printr-un singur spațiu, care reprezintă coordonatele unui punct (perechea (I, J)), respectiv direcțiile pe care acest punct poate fi legat de vecinii direcți.

Codificarea direcțiilor se face printr-un număr cuprins între 0 și 255. Reprezentarea binară a acestuia, pe 8 cifre semnifică, începând de la stânga spre dreapta, legătură pe direcțiile: *N NE E SE S SV V NV*. Valoarea 0 indică faptul că punctul nu poate fi conectat de vecinul direct de pe direcția respectivă.

De exemplu:

$(1\,0\,0\,0\,0\,1\,1\,0)_2 = 134_{10}$, deci, legături spre *N*, *SV*, *V*.

Date de ieșire

Fișierul de ieșire **`poligon.out`** va conține pe o singură linie un singur număr natural care reprezintă numărul de patrulatere care respectă condiția din enunț.

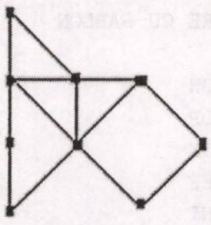
Restricții

- $1 < M, N \leq 200$;
- $4 \leq K \leq 50$.

Exemplu

`poligon.in`
```
4 4 9
1 1 24
2 1 184
2 2 43
2 3 22
3 1 136
3 2 213
3 4 5
4 1 192
4 3 65
```

`poligon.out`
```
6
```



Timp de execuție: 2 secunde/test

P040414: Șablon

Gigel și *Vasilică* își imaginează un mod de a transmite mesaje pe care nimeni să nu le poată descifra.

Mesajul este ascuns într-un text care are N linii și pe fiecare linie sunt exact N caractere - litere mari ale alfabetului englez, cifre, semne de punctuație și caracterul spațiu.

Decodificarea se face cu ajutorul unui șablon, de aceleași dimensiuni ca și textul, care are câteva găuri. Suprapunând șablonul peste text rămân vizibile câteva caractere. Acestea se citesc în ordinea liniilor, de sus în jos, iar pe aceeași linie de la stânga la dreapta. Apoi hârtia cu textul se rotește spre stânga, în sens trigonometric, cu 90°, șablonul rămânând fix. Alte caractere devin vizibile și acestea se citesc în același mod. Operația se repetă de încă două ori (rotire cu 180°, respectiv cu 270°), până când textul ajunge, printr-o nouă rotație cu 90°, din nou în poziția inițială.

Din păcate, șablonul pentru codificare/decodificare s-a pierdut. În schimb a rămas la *Gigel* mesajul inițial, iar la *Vasilică* a ajuns textul care conține mesajul.

Se cere să se reconstituie șablonul care a fost folosit la codificare.

Date de intrare

Fișierul de intrare **sablon.in** conține pe prima linie mesajul inițial. Pe linia a doua a fișierului de intrare se găsește valoarea *N*. Următoarele *N* linii conțin textul care ascunde mesajul.

Date de ieșire

Fișierul de ieșire **sablon.out** trebuie să conțină *N* linii a câte *N* caractere. Caracterele pot fi "o" (pentru reprezentarea unei găuri) și "x".

Restricții și precizări

- prin rotirea textului nici una din găuri nu se va suprapune peste nici una din pozițiile ocupate de o gaură în pozițiile precedente ale textului;
- $1 \le N \le 50$.
- mesajul are cel mult 1000 caractere și se încheie cu un caracter diferit de spațiu;
- în cazul în care există mai multe soluții, afișați una dintre ele.

Exemplu

sablon.in

```
CODIFICARE CU SABLON
10
ABCDCEFAGH
IJOKLEMNOP
DQRSTUVWCX
YZAIBCRDEF
GFHIJKLMNI
AJKLMNOPSQ
RSTOUV WXY
ZBABCDEFGU
HIJKNLMCNO
PQLRS TUVW
```

sablon.out

```
XXXXOXXXXX
XXOXXXXXXX
OXXXXXXXXX
XXXOXXXXXX
XOXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
```

Timp de execuție: 1 secundă/test

P040415: Șir

Gigel se distrează construind șiruri crescătoare de numere din mulțimea {1, 2, ..., *n*}. La un moment dat observă că unele șiruri, de cel puțin *k* termeni ($k \ge 3$), au o proprietate mai aparte: diferența dintre doi termeni consecutivi este constantă. Iată câteva exemple de astfel de șiruri pentru $n \ge 22$:

2, 3 ,4
1, 5, 9, 13
7, 10, 13, 16, 19, 22

Dându-se numărul natural *n* ajutați-l pe *Gigel* să numere câte astfel de șiruri poate să construiască.

Date de intrare

În fișierul de intrare **sir.in** se găsește, pe prima linie, numărul *n*.

Date de ieșire

În fișierul de ieșire **sir.out** se va afișa, pe prima linie, numărul cerut urmat de caracterul sfârșit de linie.

Restricții și precizări

- dacă există mai multe soluții, se va determina una singură;
- $3 \le n \le 20.000$;
- $3 \le k \le n$.

Exemple

sir.in	**sir.out**
3	1

sir.in	**sir.out**
4	3

sir.in	**sir.out**
5	7

Timp de execuție: 1 secundă/test

P040416: Snipers

Se spune că în timpul războiului cu *gnomii*, *trolii* au trimis *n* trăgători de elită să lichideze cele *n* căpetenii inamice.

Din fericire căpeteniile inamice erau plasate în câmp deschis, iar trăgătorii au reușit să se plaseze în zonă fără să fie observați. Când să fie dată comanda de tragere s-a constatat că nu se transmisese fiecărui trăgător ce căpetenie să împuște, iar dacă doi trăgători ar fi tras în aceeași căpetenie

sau traiectoriile razelor ucigașe s-ar fi intersectat, atunci ar fi scăpat cel puțin o căpetenie care ar fi putut duce războiul până la capăt, iar *trolii* ar fi fost învinși. Deoarece căpeteniile aveau capacitatea de a deveni invizibile oricând doreau (pe o perioadă nelimitată), trebuiau lichidate simultan, altfel... Istoria ne spune că *trolii* au învins deoarece comandantul lor a reușit ca în mai puțin de o secundă să transmită fiecărui trăgător în ce căpetenie să tragă. Voi puteți face asta?

Scrieți un program care, citind pozițiile trăgătorilor și a căpeteniilor, determină căpetenia în care trebuie să tragă fiecare trăgător.

Date de intrare

Fișierul de intrare **snipers.in** conține pe prima sa linie numărul n. Pe următoarele n linii se află perechi de numere întregi, separate prin spațiu, care reprezintă coordonatele trăgătorilor urmate de alte n perechi de numere întregi care reprezintă coordonatele căpeteniilor (abscisă și ordonată).

Date de ieșire

Fișierul de ieșire **snipers.out** va conține n linii. Linia i a fișierului va conține numărul căpeteniei țintite de trăgătorul i ($1 \le i \le n$).

Restricții și precizări

- $0 < n < 200$;
- coordonatele sunt numere întregi cuprinse între 0 și 50000;
- raza ucigașă a oricărei arme se oprește în ținta sa;
- în datele de intrare nu vor exista trei persoane aflate în puncte coliniare.

Exemple

snipers.in

```
2
1 3
1 1
3 4
3 1
```

snipers.out

```
1
2
```

snipers.in

```
5
6 6
4 13
2 8
9 4
5 2
6 11
9 7
3 9
1 4
7 3
```

snipers.out

```
2
1
3
4
5
```

Timp de execuție: 1 secundă/test

Clasa a X-a

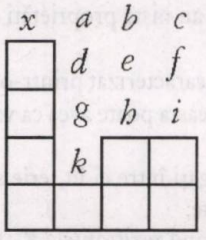
P040417: Găina

Găina *Chucky* 767 trebuie să străbată curtea sărind de pe un coteț pe altul, sau zburând peste cotețe, de la primul până la ultimul coteț. Cotețele sunt reprezentate prin niște dreptunghiuri de lățime 1 m și înălțimi date și sunt numerotate în ordine de la stânga cu numere de 1 la n. Două cotețe cu numere consecutive sunt lipite (adiacente). Inițial, *Chucky* are un număr de unități de energie. Dacă la un moment dat *Chucky* ajunge să aibă energie strict negativă sau se află în imposibilitatea de a mai face un pas (întâlnește un coteț mai înalt decât cota la care se află), atunci nu mai poate continua acel drum.

Chucky poate să se deplaseze făcând următoarele tipuri de mișcări:

- *pas* – găina pășește pe orizontală de pe un coteț de înălțime H pe următorul coteț de aceeași înălțime (în desenul de mai jos: de la poziția h la i); în acest caz nu se pierde și nu se câștigă energie;
- *aterizare* – găina aterizează pe un coteț de înălțime H, venind în zbor, tot de la înălțimea H (exemplu desenul de mai jos: de la g la h); nici în acest caz nu se pierde și nu se câștigă energie;
- *decolare* – găina zboară 1 m pe orizontală de pe un coteț (în desen de la x la a); în acest caz găina pierde o unitate de energie;
- *zbor orizontal* – găina zboară pe orizontală (exemplu în desen, de la a la b, de la b la c, ...); în acest caz pierde câte o unitate de energie pentru fiecare metru parcurs pe orizontală.
- *picaj* – găina coboară pe verticală (exemplu în desen de la a la d, sau de la d la g ...); în acest caz câștigă câte o unitate de energie pentru fiecare metru coborât.

Mai jos, avem un drum format din patru cotețe, de înălțimi 4, 1, 2, 2. Exemplificăm diferite tipuri de mișcări din poziția x (ceea ce nu reprezintă un traseu complet).



În figura anterioară, din poziția de start x se poate ajunge în poziția h pe 3 trasee:

- x, a, b, e, h cu energia inițială 2:
  $x \to a(-1), a \to b(-1), b \to e(+1), e \to h(+1)$
- x, a, b, e, h cu energia inițială 1:
  $x \to a(-1), a \to d(+1), d \to e(-1), e \to h(+1)$
- x, a, d, g, h cu energia inițială 1:
  $x \to a(-1), a \to d(+1), d \to g(+1), g \to h(0)$

Să se determine energia minimă (un număr natural) pe care trebuie să o aibă *Chucky* la începutul călătoriei (când se află pe primul coteț), astfel încât ea să poată ajunge pe cotețul n, având în fiecare moment energia mai mare sau egală cu 0.

Date de intrare

Fișierul de intrare **gaina.in** conține două linii. Pe prima linie se află numărul natural n. Pe linia a doua se află n numere naturale $h_1, h_2, ..., h_n$ (unde h_i reprezintă înălțimea cotețului i), separate printr-un spațiu.

Date de ieșire

Fișierul de ieșire **gaina.out** va conține o singură linie pe care se află un număr natural K reprezentând energia minimă inițială cu care găina *Chucky* poate să ajungă la cotețul n, având în fiecare moment energia mai mare sau egală cu 0.

Restricții și precizări

- $0 < n < 10001$;
- $0 \le h_i \le 30000$;
- se garantează existența unei soluții (primul coteț are înălțimea mai mare sau egală cu înălțimea oricărui alt coteț).

Exemple

gaina.in
```
3
2 2 0
```

gaina.out
```
16
```

gaina.in
```
6
3 0 0 2 1 1
```

gaina.out
```
2
```

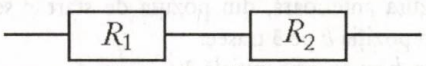
Timpi de execuție:

- *Microsoft Windows*: 0,1 secunde/test
- *Linux*: 0,1 secunde/test

P040418: Rez

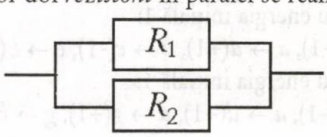
Gigel este electronist amator. El afirmă că a inventat o nouă componentă electronică denumită *reziston*. În mod ciudat totuși *rezistonii* au niște proprietăți care nouă ne sună foarte cunoscut:

- orice *reziston* este caracterizat printr-o mărime fizică numită *rezistență*; aceasta poate avea ca valori numai numere naturale;
- *rezistonii* pot fi legați între ei în serie sau în paralel, formând astfel circuite;
- fie doi *rezistoni* având *rezistențele* R_1, respectiv R_2; legarea în serie a *rezistonilor* se realizează astfel:



Rezistența acestui circuit va fi $R_1 + R_2$.

- legarea celor doi *rezistoni* în paralel se realizează astfel:



Rezistența acestui circuit va fi $(R_1 \cdot R_2) / (R_1 + R_2)$. Fiindcă *rezistențele* pot fi numai numere naturale, împărțirea este întreagă (adică rezultatul este câtul împărțirii întregi a lui $(R_1 \cdot R_2)$ la $(R_1 + R_2)$).

- prin legarea oricâtor *rezistoni* în serie și în paralel se obțin circuite; circuitele pot fi legate în serie și/sau în paralel după aceleași reguli; *rezistența* unui circuit se calculează aplicând regulile de mai sus.

Un circuit va fi codificat printr-un șir de caractere construit după următoarele reguli:

- dacă circuitul C este format dintr-un singur *reziston* și acesta are *rezistența* de valoare x, atunci codificarea circuitului C este Rx; *rezistența* circuitului C va fi x.
- dacă circuitul C este obținut prin legarea în serie a două sau mai multe circuite, codificate $C_1, C_2, ..., C_k$, atunci codificarea circuitului C se obține concatenând în ordine codificările circuitelor $C_1C_2...C_k$; *rezistența* circuitului C se obține prin însumarea *rezistențelor* circuitelor $C_1, C_2, ..., C_k$;
- dacă circuitul C este obținut prin legarea în paralel a două sau mai multe circuite, atunci codificarea circuitului C se obține încadrând între paranteze rotunde codificările circuitelor din care este format și separând aceste codificări prin virgulă: $(C_1, C_2, ..., C_k)$, $k > 1$; *rezistența* circuitului C este egală cu câtul împărțirii produsului dintre *rezistențele* $C_1, C_2, ..., C_k$ și suma *rezistențelor* circuitelor $C_1, C_2, ..., C_k$.

Scrieți un program care să determine *rezistența* unui circuit.

Date de intrare

Fișierul de intrare **rez.in** conține pe prima linie un șir de caractere care reprezintă codificarea unui circuit conform regulilor de mai sus.

Date de ieșire

Fișierul de ieșire **rez.out** conține o singură linie pe care este scrisă *rezistența* circuitului specificat în fișierul de intrare.

Restricții și precizări

- lungimea codificării unui circuit este un număr întreg cuprins între 0 și 1000, exclusiv;
- *rezistența* oricărui *reziston* este un număr întreg cuprins între 0 și 100, exclusiv;
- *rezistența* oricărui circuit este un număr întreg cuprins între 0 și 2.000.000.000, exclusiv;
- șirul prin care se codifică un circuit nu conține spații;
- pentru datele de test nu se vor obține împărțiri la 0.

Exemple

rez.in
```
R12
```

rez.out
```
12
```

rez.in
```
R42R33R3
```

rez.out
```
78
```

rez.in	rez.out
R2(R5,R69,R12)R80	130

rez.in	rez.out
(R5,R3(R12,R4),R3)	6

Timpi de execuție:
- *Microsoft Windows*: 0,1 secunde/test
- *Linux*: 0,1 secunde/test

P040419: Sortări

Balaurului Arhirel nu îi plac prea mult șirurile care nu sunt ordonate. Din acest motiv, nu poate să suporte permutările de N elemente, așa că se decide să le sorteze și pentru asta inventează o metodă proprie de sortare.

El ia inițial un șir S care reprezintă o permutare de ordin N. Caută în șirul S cel mai mic (*min*) și cel mai mare element (*max*). Să considerăm că *min* se află în șirul S pe poziția *pmin*, iar *max* pe poziția *pmax*. Să notăm cu x minimul dintre *pmin* și *pmax*, iar cu y maximul dintre *pmin* și *pmax*. Șirul S a fost astfel partiționat în alte trei șiruri S_1, S_2, S_3 care pot avea fiecare zero elemente, un element sau mai multe elemente. Șirul S_1 începe la prima poziție din șir și se termină la poziția x - 1. Șirul S_2 începe la poziția x + 1 și se termină la poziția y - 1. Șirul S_3 începe la poziția y + 1 și se termină la ultima poziție din șir.

Balaurul Arhirel mută valoarea *min* la capătul din stânga al lui S, iar valoarea *max* la capătul din dreapta al șirului S și reia sortarea pentru fiecare din șirurile S_1, S_2 și S_3.

De exemplu, să considerăm N = 6 și șirul S = (3 4 2 1 6 5). La primul pas *min* = 1 și *max* = 6. Deci S_1 = (3 4 2); S_2 = (); S_3 = (5). Se mută *min* și *max* la capetele șirului și se obține S = (1 3 4 2 5 6) și se sortează în același mod S_1, S_2 și S_3. S_2 și S_3 au 0, respectiv 1 element, deci sunt deja sortate. Pentru S_1, se găsește *min* = 2 și *max* = 4 și vom avea șirurile (3), () și (). Se mută *min* și *max* la capete și se obține S_1 = (2 3 4). În final, vom avea șirul S = (1 2 3 4 5 6).

Evident, această metodă nu va funcționa întotdeauna pentru sortarea unei permutări. Spre exemplu, pentru șirul S = (3 4 1 6 5 2), se găsește *min* = 1 și *max* = 6, iar S_1 = (3 4), S_2 = (), S_3 = (5 2). Se mută *min* și *max* la capetele lui S: S = (1 3 4 5 2 6) și se procedează la sortarea pe rând a șirurilor S_1, S_2, S_3. S_1 este sortat, S_2 nu are elemente, iar S_3 va deveni S_3 = (2 5). În final, S = (1 3 4 2 5 6).

Ajutați-l pe *Balaurul Arhirel* să afle câte dintre permutările de N elemente pot fi sortate prin metoda sa.

Date de intrare

Fișierul **sortari.in** conține o singură linie pe care se află numărul N.

Date de ieșire

Fișierul **sortari.out** va conține o singură linie pe care se află numărul de permutări de ordin N ce pot fi sortate prin metoda balaurului modulo 19573 (restul împărțirii numărului de permutări la 19573).

Restricție

- $0 < N < 201$.

Exemplu

sortari.in	sortari.out
4	18

Explicație

În cazul permutărilor de câte patru elemente, șirurile (1 3 4 2); (3 1 2 4); (3 1 4 2); (3 4 1 2); (3 4 2 1); (4 3 1 2) nu vor putea fi sortate crescător. Celelalte 24 - 6 = 18 permutări pot fi sortate crescător.

De exemplu, pentru șirul (1 3 4 2), după găsirea *min* = 1, *max* = 4, se obțin șirurile: S_1 = (); S_2 = (3); S_3 = (2), care, având câte un element sunt sortate. Astfel, după mutarea la capete a lui *min* și *max* vom avea: (1 3 2 4)

Timpi de execuție:
- *Microsoft Windows*: 1,5 secunde/test
- *Linux*: 0,2 secunde/test

P040420: Cuvinte

Balaurul Arhirel se decide să învețe biologie, așa că dorește să cumpere manualul de clasa a X-a. Din păcate, acesta nu se mai găsește pe piață, dar *Balaurul* reușește să găsească o copie la un prieten.

După ce începe să citească, *Balaurul Arhirel* observă că există greșeli în copia prietenului, iar într-un impuls de energie, se hotărăște să corecteze manualul.

El are la dispoziție un dicționar de M cuvinte dintre care trebuie să extragă variante pentru cuvântul greșit. Asupra cuvântului greșit balaurul poate să facă următoarele schimbări în așa fel încât acesta să ajungă la o variantă din dicționar:
- poate șterge o literă;
- poate insera o literă;
- poate schimba o literă în altă literă.

Totuși, *Balaurul Arhirel* este leneș, așa că nu dorește să opereze mai mult de K schimbări în cuvântul greșit pentru a-l aduce la o formă corectă (existentă în dicționar).

Ajutați-l pe *Balaurul Arhirel* să afle care dintre cuvintele din dicționar ar putea fi variante ale cuvântului greșit.

Date de intrare

Fișierul de intrare **cuvinte.in** conține pe prima linie cele două numere M și K, separate printr-un spațiu, reprezentând numărul de cuvinte din dicționar și numărul maxim de modificări care pot fi efectuate asupra cuvântului care trebuie corectat.

Pe cea de-a doua linie se găsesc separate printr-un spațiu lungimea cuvântului greșit, $L_{cuvânt}$, și cuvântul greșit.

Pe următoarele M linii se găsesc cuvintele din dicționar, câte un cuvânt pe o linie în forma următoare: pe linia i lungimea L_{i-2} a cuvântului i - 2, separată printr-un singur spațiu de cuvântul i - 2.

Date de ieșire

Fișierul de ieșire **cuvinte.out** va conține M linii. Pe linia i se află valoarea 1 pentru cazul în care cuvântul i din dicționar este o variantă pentru cuvântul greșit dat, respectiv valoarea 0 în caz contrar.

Restricții și precizări

- $0 < M < 21$;
- $0 < K < 31$;
- lungimea oricărui cuvânt (inclusiv greșit) este un număr întreg cuprins între 0 și 10001, exclusiv;
- cuvintele sunt formate doar din literele alfabetului latin, iar literele mici diferă de cele mari (de exemplu, z nu este același lucru cu z).

Exemplu

cuvinte.in	cuvinte.out
6 2 6 radiux 5 ladin 6 Radius 6 ridica 5 radio 6 adipos 5 cadiu	0 1 0 1 0 1

Timpi de execuție:

- *Microsoft Windows*: 2 secunde/test
- *Linux*: 0,3 secunde/test

P040421: Materom

Liceul Național Anonim (*LNA*) este invitat să participe la olimpiada de matematică-română cu o echipă formată din m elevi.

La această olimpiadă elevii lucrează în echipă și trebuie să rezolve două subiecte: unul de română și altul de matematică.

Au fost testați și punctați la cele două materii n elevi, numerotați de la 1 la n.

Așa cum era de așteptat, în general, elevii buni la matematică s-au dovedit a fi cam slăbuți la română și viceversa.

Pentru a maximiza șansele de câștig ale echipei *LNA*, directorul a decis să trimită m elevi dintre cei n elevi testați, astfel încât diferența în modul dintre suma punctajelor de la limba română ale elevilor din echipă și suma punctajelor la matematică ale elevilor din echipă să fie minimă. Dacă există mai multe echipe de elevi care îndeplinesc condiția precedentă, va fi selectată dintre acestea o echipă pentru care suma tuturor notelor să fie maximă.

Scrieți un program care să determine în conformitate cu decizia directorului, diferența în modul dintre suma punctajelor de la limba română ale elevilor din echipa *LNA* și suma punctajelor la matematică ale elevilor din echipă, precum și suma tuturor punctajelor elevilor din echipa *LNA*.

Date de intrare

În fișierul de intrare **materom.in** se află pe prima linie numerele naturale n și m separate printr-un spațiu, având semnificația din enunț.

Pe fiecare dintre următoarele n linii se află două numere naturale separate printr-un spațiu. Mai exact linia i din fișier ($2 \le i \le n + 1$) conține m_i și r_i, unde m_i este punctajul obținut la matematică, iar r_i este punctajul obținut la limba română de elevul i - 1.

Date de ieșire

Fișierul de ieșire **materom.out** conține două linii. Pe prima linie se va afișa diferența (în modul) dintre suma punctajelor de la limba română ale elevilor din echipă și suma punctajelor la matematică ale elevilor din echipă. Pe cea de a doua linie se va afișa suma punctajelor elevilor selectați în echipa *LNA*.

Restricții și precizări

- $1 \le m < 20$;
- $1 \le n \le 500$;
- $m \le n$;
- $0 \le m_i, r_i \le 20$.

Exemplu

materom.in	materom.out
4 2 2 3 1 2 6 2 4 1	2 10

Explicație

Dintre cei patru elevi trebuie să selectăm doi. Avem 6 posibilități, dintre care trei au diferența (în modul) dintre suma notelor la matematică și suma notelor la română 2.

Acestea sunt:

- (1, 2) pentru care suma punctajelor este 8;
- (1, 4) pentru care suma punctajelor este 10;
- (2, 4) pentru care suma punctajelor este 8.

Alegem combinația 1 4, deoarece are suma maximă.

Timpi de execuție:

- *Microsoft Windows*: 0,2 secunde/test
- *Linux*: 0,1 secunde/test

P040422: Puncte

Considerăm că toate punctele de coordonate întregi din plan sunt colorate în negru, cu excepția a n puncte care sunt colorate în roșu. Două puncte roșii aflate pe aceeași linie orizontală sau pe aceeași linie verticală (adică puncte care au aceeași ordonată sau aceeași abscisă) pot fi unite printr-un segment. Colorăm în roșu toate punctele de coordonate întregi de pe acest segment. Repetăm operația cât timp se obțin puncte roșii noi.

Cunoscând coordonatele celor n puncte care erau inițial roșii, aflați numărul maxim de puncte roșii care vor exista în final.

Date de intrare

Pe prima linie a fișierului de intrare **puncte.in** se află numărul n. Pe următoarele n linii sunt date coordonatele punctelor, separate printr-un singur spațiu.

Date de ieșire

Fișierul de ieșire **puncte.out** va conține o singură linie pe care se află numărul maxim de puncte roșii existente în final.

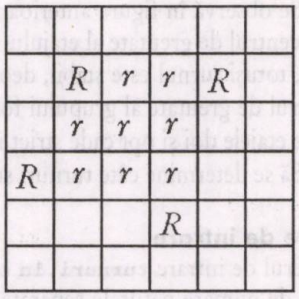
Restricții și precizări

- $0 \le n \le 100.000$;
- coordonatele sunt numere întregi cuprinse între 0 și 1000.

Exemplu

puncte.in

```
4
0 2
3 1
1 4
4 4
```



puncte.out

```
12
```

Timpi de execuție:

- *Microsoft Windows*: 0,1 secunde/test
- *Linux*: 0,1 secunde/test

Clasele a XI-a și a XII-a

P040423: Color

Ion și *Vasile* joacă un joc. Ei au la dispoziție un arbore binar strict (adică fiecare nod are 0 sau doi fii) cu N noduri, numerotate de la 1 la N (nodul numerotat cu 1 este rădăcina arborelui). Inițial, toate nodurile sunt colorate în alb. Jucătorii vor efectua mutări alternativ, iar jucătorul aflat la mutare va colora în negru un nod colorat în alb. *Ion* efectuează prima mutare și poate colora în negru orice nod al arborelui. Considerând că ultimul nod colorat de unul dintre jucători este P, jucătorul care urmează la mutare poate colora în negru unul din următoarele noduri (dacă nu au fost deja colorate în negru):

- unul din cei doi fii ai lui P (dacă P nu este frunză în arbore);
- tatăl lui P (dacă P nu este rădăcina arborelui).

Jocul continuă până când unul dintre jucători nu mai poate efectua nici o mutare. Atunci, jucătorul care a efectuat ultima mutare este considerat câștigător.

Considerând că ambii jucători joacă optim, determinați toate nodurile din arbore pe care le poate colora *Ion* la prima mutare, astfel încât să fie sigur de victorie.

Date de intrare

Prima linie a fișierului **color.in** conține numărul întreg N, reprezentând numărul de noduri din arbore.

Următoarele N - 1 linii conțin câte două numere întregi separate printr-un spațiu, a și b, având semnificația că a este tatăl lui b.

Date de ieșire

Pe prima linie a fișierului **color.out** se va afla numărul întreg M, reprezentând numărul de noduri pe care le poate colora *Ion* la prima mutare, astfel încât să fie sigur de victorie. Pe următoarea linie veți afișa numerele acestor noduri, în ordine crescătoare.

Restricție

- $1 \le N \le 16.000$, N impar.

Exemplu

color.in

```
9
1 2
1 3
2 4
2 5
4 6
4 7
3 8
3 9
```

color.out

```
6
1 5 6 7 8 9
```

Timpi de execuție:

- *Microsoft Windows*: 0,2 secunde/test
- *Linux*: 0,3 secunde/test

P040424: Magic

Misqpan și *Trofonaced* sunt doi eroi care vor să-și unească forțele în lupta împotriva răului. Regatul este reprezentat printr-o matrice dreptunghiulară de N linii și M coloane. Fiecare element al matricei corespunde unei bucăți de teren uscat sau mlăștinos. Cei doi eroi nu se vor aventura în părțile mlăștinoase ale regatului – se vor deplasa numai pe uscat. Ei se pot muta dintr-o poziție a matricei în una din cele patru poziții vecine pe orizontală sau pe verticală, dacă această poziție corespunde unei zone de uscat. Unele poziții de uscat pot fi transformate prin vrajă în mlaștină.

Ajutați un vrăjitor malefic să aleagă un număr minim de poziții "transformabile", prin schimbarea cărora cei doi eroi să nu se poată întâlni (să nu existe un drum pe uscat între cei doi).

Date de intrare

Prima linie a fișierului **magic.in** conține două numere întregi N și M, reprezentând numărul de linii, respectiv de coloane ale matricei. Următoarele N linii conțin câte M caractere cu următoarea semnificație:

- . – poziție uscată;
- x(mic) – poziție mlăștinoasă;

- * – poziție uscată "transformabilă" în una mlăștinoasă de către vrăjitor;
- M – poziția eroului *Misopan*;
- T – pentru poziția eroului *Trofonaced*.

Date de ieșire

Pe prima linie a fișierului **magic.out** se scrie numărul întreg R, reprezentând numărul minim de poziții care trebuie transformate. Pe următoarele R linii vor apărea câte două numere, reprezentând pozițiile alese. Primul număr va fi linia (între 1 și N), iar al doilea va fi coloana (între 1 și M).

Restricții

- $1 \le N, M \le 50$;
- R, rezultatul afișat, poate fi 0;
- se garantează existența unei soluții;
- se garantează că în toată matricea caracterele M, respectiv T vor apărea fiecare exact o dată;
- pozițiile eroilor sunt implicit zone de uscat care nu pot fi transformate de vrăjitor.

Exemplu

magic.in	**magic.out**
4 4	1
MxxT	3 3
.x*.	
.**.	
**x.	

Timpi de execuție:

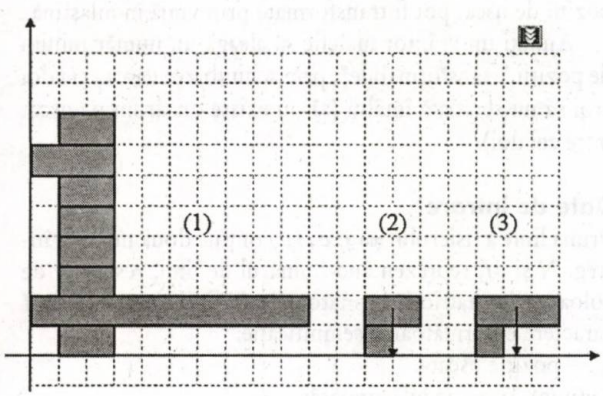
- *Microsoft Windows*: 1,8 secunde/test
- *Linux*: 0,6 secunde/test

P040425: Turnuri

Renumitul arhitect *Prăbușilă* dorește să construiască unul dintre cele mai interesante turnuri de pe planetă. Acest turn, în mod cu totul deosebit, va avea etaje de diverse lățimi, între 1 și 100, numere întregi.

Prăbușilă s-a hotărât deja ce dimensiune va avea fiecare dintre etajele turnului, dar nu și cum să le așeze pe orizontală. El ar dori mai întâi să știe câte variante are.

Dacă cele două informații sunt egale, căutarea se termină cu succes.



Etajele pot fi așezate la coordonate întregi și va trebui ca un astfel de turn să nu se dărâme.

- condiția pentru ca un turn să fie stabil este ca la fiecare etaj perpendiculara coborâtă din centrul de greutate al grupului etajelor superioare să cadă strict în interiorul acelui etaj (nu are voie să fie pe margini sau în afară - de exemplu, turnurile 2 și 3 sunt instabile);
- centrul de greutate al unui etaj se află la mijlocul etajului respectiv;
- centrul de greutate al unui grup de etaje are drept coordonată x (orizontală) media coordonatelor x ale centrelor de greutate ale etajelor componente. (Etajele au mase egale, indiferent de cât de late sunt).

În figura anterioară, etajul din vârf are coordonata x a centrului de greutate 2, iar grupul celor două etaje din vârf are centrul de greutate la coordonata $x = 1{,}75$ (media aritmetică dintre 2 și 1,5)

Se observă în figura anterioară că, deși perpendiculara din centrul de greutate al etajului doi cade în afara etajului unu, totuși turnul este stabil, deoarece perpendiculara din centrul de greutate al grupului format din etajele cuprinse între etajele doi și opt cade strict în interiorul etajului unu.

Să se determine câte turnuri stabile există.

Date de intrare

Fișierul de intrare **turnuri.in** conține pe o singură linie lista de numere naturale separate prin câte un spațiu, numere reprezentând lățimile etajelor turnului, începând cu cel mai de sus. Lista se termină cu un 0.

Date de ieșire

Fișierul de ieșire **turnuri.out** trebuie să conțină numărul de turnuri.

Restricții și precizări

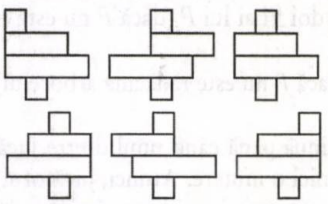
- numărul maxim de turnuri nu va depăși 2.000.000.000;
- numărul maxim de etaje ale unui turn este 200;
- lățimea maximă a unui etaj este 100.

Exemplu

turnuri.in	**turnuri.out**
1 3 4 1 0	6

Explicație

Cele șase variante sunt:



Timpi de execuție:

- *Microsoft Windows*: 0,2 secunde/test
- *Linux*: 0,2 secunde/test

P050326: Coach

Sunteți antrenorul ciclistului *Adirem Onamihs*. În curând va avea loc un eveniment sportiv, iar pentru organizarea acestuia s-au amenajat N intersecții și M drumuri bidirecționale între aceste intersecții. Pentru fiecare drum se cunoaște numărul de minute necesare pentru parcurgerea lui. La fiecare intersecție ciclistul care trece pe acolo este obligat să servească o băutură energizantă și răcoritoare. Băutura diferă de la intersecție la intersecție și se cunoaște deja numărul de calorii ale fiecărei băuturi.

Ca mare antrenor, urmează să întocmiți un plan special pentru a-l antrena pe *Adirem*. Doriți ca durata traseului pe care îl alege *Adirem* să aibă exact T minute, însă nu vreți să-i plănuiți întregul traseu (*Adirem* trebuie să își antreneze și mintea, nu numai corpul). Îi veți preciza lui *Adirem* intersecția de unde își începe traseul și intersecția unde îl termină. *Adirem* învață repede – el știe întotdeauna să aleagă traseul optim (drumul cel mai scurt dintre cele două intersecții). Pentru a-l face să meargă exact T minute îi veți interzice trecerea prin anumite intersecții, sub pretextul că valoarea calorică a băuturii servite în intersecția respectivă nu este benefică pentru antrenamentul lui. Astfel, îi veți preciza o limită inferioară și una superioară pentru numărul de calorii ale băuturilor pe care el are voie să le bea. *Adirem* nu va trece decât prin intersecțiile unde se servește o băutură cu valoare calorică între limitele date.

Scrieți un program care să calculeze cele patru variabile din antrenamentul lui *Adirem*: intersecția de start, intersecția de sfârșit, valoarea calorică minimă pe care poate să o consume și valoarea calorică maximă, astfel încât drumul cel mai scurt dintre cele două intersecții (care să respecte restricțiile) să dureze exact T minute.

Date de intrare

Prima linie a fișierului `coach.in` conține trei numere întregi N, M și T – numărul de intersecții, numărul de drumuri, respectiv timpul dorit. Următoarele N linii conțin câte un număr – valorile calorice (întregi între 1 și 10000 inclusiv) ale băuturilor din intersecții, în ordine (de la 1 la N). Următoarele M linii conțin câte un triplet de numere: două intersecții (numere distincte între 1 și N) și durata drumului dintre ele (întreg între 1 și 10000 inclusiv).

Date de ieșire

Fișierul `coach.out` va conține o linie pe care se vor afla cele patru valori găsite: nodul de start, nodul de sfârșit, valoarea calorică minimă și valoarea calorică maximă. Nodurile vor fi întregi între 1 și N, iar valorile calorice vor fi întregi între 1 și 10000 (inclusiv).

Restricții și precizări

- $1 \le N \le 200$;
- $1 \le M \le 4950$;
- $1 \le T \le 1.000.000$;
- intersecțiile găsite (de start și de oprire) trebuie să respecte și ele restricțiile calorice;
- o băutură cu valoare calorică x poate fi băută dacă și numai dacă $c_{min} \le x \le c_{max}$, unde c_{min} și c_{max} sunt valorile calorice minime și maxime stabilite de antrenor;
- între două intersecții există maximum un drum;
- valorile calorice sunt distincte;
- există întotdeauna soluție; dacă există mai multe soluții se cere oricare dintre ele.

Exemplu

`coach.in`

```
6 9 11
40
10
20
30
60
50
1 2 2
1 3 2
1 4 4
1 6 10
2 3 3
2 4 1
4 5 1
4 6 5
5 6 2
```

`coach.out`

```
3 6 20 55
```

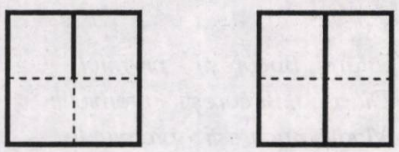
Timpi de execuție:

- *Microsoft Windows*: 0,9 secunde/test
- *Linux*: 0,5 secunde/test

P040427: Pătrate

Ovi este un băiețel foarte isteț căruia îi place să scrie pe asfalt cu creta și să țopăie. El desenează cu cretă roșie un dreptunghi de lățime exact 2 metri și lungime N metri, pe care îl împarte în pătrate egale de latură 1 metru, unele laturi interioare fiind desenate cu cretă roșie, iar restul laturilor interioare cu cretă albă. *Ovi* pornește din pătratul aflat în colțul stânga-sus al dreptunghiului, sărind dintr-un pătrat în altul vecin pe linie sau coloană, cu condiția ca latura care desparte cele două pătrate să nu fie colorată în roșu. El își dorește ca prin sărituri succesive să ajungă în toate pătratele dreptunghiului, dar a observat că numai pentru anumite variante de colorare a laturilor pătratelor reușește acest lucru.

În exemplele de mai jos (cu $N = 2$) liniile interioare îngroșate sunt colorate cu roșu, iar cele punctate sunt colorate cu alb. În exemplul din stânga din figura următoare, pornind din colțul stânga-sus se poate ajunge în oricare alt pătrat, dar în exemplul din dreapta din figura următoare nu se poate ajunge la pătratele din partea dreaptă.



Ajutați-l pe *Ovi* să numere câte posibilități de colorare în roșu a unor laturi interioare ale pătratelor sunt, astfel încât plecând din colțul stânga-sus să poată ajunge prin sărituri în oricare alt pătrat.

Date de intrare

În fișierul **patrate.in** se află un singur număr natural N care reprezintă lungimea în metri a dreptunghiului.

Date de ieșire

În fișierul **patrate.out** se va scrie un singur număr natural (urmat de caracterul de sfârșit de linie) care reprezintă numărul de posibilități cerut.

Restricție

- $1 \leq N \leq 1000$.

Exemplu

patrate.in

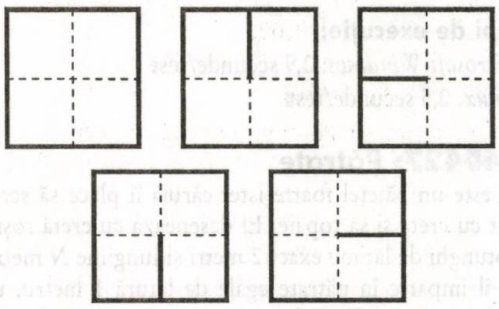
```
2
```

patrate.out

```
5
```

Explicație

Cele cinci posibilități sunt:



Timpi de execuție:

- *Microsoft Windows*: 0,4 secunde/test
- *Linux*: 0,2 secunde/test

P040428: Base3

Se consideră trei numere scrise în baza 3 (folosind cifrele 0, 1 și 2). Se dorește găsirea unui număr N în baza 3, care să aibă un număr impar de cifre, iar cifra de pe poziția din mijloc să aibă valoarea 1. Acest număr N trebuie obținut prin concatenarea celor trei numere date; în această concatenare, fiecare dintre cele 3 numere poate fi folosit de zero sau mai multe ori.

Determinați numărul minim de cifre pe care îl poate avea un număr având proprietățile precizate mai sus.

Date de intrare

Fișierul de intrare **base3.in** conține trei linii. Pe fiecare linie se află scris un număr în baza 3.

Date de ieșire

Fișierul de ieșire **base3.out** va conține numărul minim de cifre pe care îl poate avea un număr N cu proprietățile specificate. Dacă nu se poate obține nici un astfel de număr, se va afișa în fișier valoarea 0.

Restricții

- numărul de cifre al fiecăruia dintre cele trei numere este un număr întreg cuprins între 1 și 16000;
- numerele date pot conține zerouri la început; acestea trebuie luate în considerare, dacă numărul respectiv este folosit în concatenare.

Exemplu

base3.in

```
001
020
2020
```

base3.out

```
13
```

Explicație

Se poate obține numărul 2020**001**001**001**.

Timpi de execuție:

- *Microsoft Windows*: 0,8 secunde/test
- *Linux*: 0,4 secunde/test

Câștigătorii

Clasa a IX-a

Andrei Ioniță, Suceava - premiul I
Bogdan Ionescu, București - premiul II
Lucian Stănescu, Constanța - premiul III
Teodor Roșu, Vrancea - premiul III
Andrei Gönczi, Arad - premiul III

Clasa a X-a

Dan Spătărel, București - premiul I
Adrian Diaconu, București - premiul II
Adrian Vladu, București - premiul II

Clasa a XI-a

Dan-Leonard Crestez, Brăila - premiul I
Dan-Ionuț Fechete, Suceava - premiul II
Sorin Stancu-Mara, București - premiul III

Clasa a XII-a

Andrei Matei, București - premiul I
Andrei Homescu, Gorj - premiul II
Dan Ghinea, București - premiul III

28-29 februarie 2004

Faza JUDEȚEANĂ a olimpiadei de INFORMATICĂ pentru LICEE

Mihai Stroe

În perioada 28 - 29 februarie a avut loc faza județeană a olimpiadei de informatică pentru licee. În continuare vă vom prezenta soluțiile problemelor propuse spre rezolvare la acest concurs.

Clasa a IX-a

P030419: Expresie

Rezolvarea problemei constă în descompunerea în factori primi a numerelor date și prelucrarea acestor factori.

Concret, se poate păstra un vector P de la 1 la 30000, în care P_i este 0 dacă i nu este prim, respectiv puterea la care apare i în descompunerea în factori primi a produsului dacă i este prim.

Valorile din P nu pot depăși $n \cdot \log_2 30000$, deci vor fi mai mici sau egale cu $5000 \cdot 14 = 70.000$; astfel, elementele vectorului vor fi de tip *longint*.

Fiecare din cele n numere este descompus în factori primi. În momentul în care se determină un factor prim F al unui număr din cele date, se incrementează P_F cu puterea la care F apare în descompunerea numărului respectiv. Nu este necesară memorarea separată a descompunerii fiecărui număr.

În final, pentru fiecare element din P se verifică dacă este multiplu de m. Dacă toate elementele îndeplinesc această condiție, expresia este un număr natural și se trece la afișare.

Se poate renunța la vectorul de 30000 de elemente, păstrându-se în locul acestuia un vector în care se memorează numerele prime mai mici decât 30000 și un vector care arată la ce puteri apar aceste numere în descompunere. Această abordare introduce în plus operații pentru a găsi indicele unui anumit număr prim; se poate folosi cu succes căutarea binară. Pe de altă parte, la descompunerea în factori primi se va testa numai împărțirea prin numere prime.

Analiza complexității

Descompunerea unui număr X în factori primi are ordinul de complexitate $O(\sqrt{x})$, dacă nu se folosește lista de numere prime.

Pasul de descompunere și completare a vectorului P are deci ordinul de complexitate $O(n \cdot \sqrt{30000})$.

Citirea datelor, verificarea dacă expresia este un număr natural și afișarea au ordinul de complexitate $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n)$, dacă facem abstracție de constanta $\sqrt{30000}$.

P030420: Reactivi

Problema se poate rezolva prin metoda *greedy*.

O variantă mai explicită a enunțului este următoarea:

"*Se consideră N intervale pe o axă. Să se aleagă un număr minim de puncte astfel încât fiecare interval să conțină cel puțin unul dintre punctele alese.*"

Facem o primă observație: pentru orice soluție optimă există o soluție cu același număr de puncte (frigidere), în care fiecare punct să fie sfârșitul unui interval. Aceasta se poate obține mutând fiecare punct spre dreapta, până când ar ajunge la sfârșitul intervalului care se termină cel mai repede, dintre intervalele care îl conțin. Se observă că noua soluție respectă restricțiile din enunț.

În continuare ne vom concentra pe găsirea unei soluții de acest tip.

Sortăm reactivii după sfârșitul intervalului. Pentru intervalul care se termină cel mai repede, alegem ultimul punct al său ca temperatură a unui frigider. Se observă că această alegere este cea mai bună, dintre toate alegerile unui punct în intervalul respectiv, în sensul că mulțimea intervalelor care conțin punctul este mai mare (conform relației de incluziune), decât mulțimea corespunzătoare oricărei alte alegeri. Acest fapt este adevărat, deoarece mutarea punctului mai la stânga nu duce la selectarea unor noi intervale.

Intervalele care conțin punctul respectiv sunt eliminate (reactivii corespunzători pot fi plasați într-un frigider), iar procesul continuă cu intervalele rămase, în același mod.

Analiza complexității

Notăm cu D numărul de temperaturi întregi din intervalul care conține temperaturile din enunț. Se observă că D este cel mult 201.

Citirea datelor de intrare are ordinul de complexitate $O(N)$.

Sortarea intervalelor după capătul din dreapta are ordinul de complexitate $O(N \cdot \log N)$.

Urmează F pași, unde F este numărul de frigidere selectate. Deoarece fiecare frigider este setat la o temperatură întreagă, $F \leq D$.

În cadrul unui pas, determinarea intervalului care se termină cel mai repede, pe baza vectorului sortat, are ordinul de complexitate $O(1)$. Eliminarea intervalelor care conțin un anumit punct (sfârșitul intervalului care se termină cel mai repede) are ordinul de complexitate $O(N)$.

Afișarea rezultatului are ordinul de complexitate $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N \cdot D + N \cdot \log N)$; deoarece în general $D > \log N$, considerăm ordinul de complexitate ca fiind $O(N \cdot D)$.

Clasa a X-a

P030421: Perle

Inițial, problema pare dificilă, dar, după examinarea transformărilor posibile, se observă că este destul de simplă.

Fie un anumit șir de perle. Presupunem că acesta a fost obținut dintr-o perlă magică și ne oprim la prima contradicție.

Să determinăm, la început, perla magică din care s-ar putea obține șirul.

Dacă șirul are lungimea 1, el se poate obține dintr-o perlă magică A.

Dacă șirul începe cu `12` și are trei perle, s-ar putea obține dintr-o perlă C.

Dacă șirul începe cu `2` și are cel puțin două perle, singura șansă de a-l obține este de a începe cu o perlă magica de tip B. Similar dacă începe cu 1 și are mai mult de trei perle.

Dacă șirul începe cu `3`, s-ar putea obține numai dintr-o perlă de tip C.

Acum, având în vedere că am stabilit singura perlă magică din care sunt șanse să se obțină șirul, vom verifica dacă sirul poate fi într-adevăr obținut.

Pentru aceasta, putem scrie câte o funcție pentru fiecare perlă magică. O astfel de funcție se aplică pe șirul de intrare și execută operații (avansări pe șir sau apeluri ale funcțiilor pentru alte tipuri de perle) în funcție de tipul perlelor întâlnite.

Dacă la un moment dat nu există nici o regulă de continuare (de exemplu, pentru șirul `22`), șirul nu poate fi obținut.

Funcția pentru A nu este necesară, tratarea perlei magice A putând fi inclusă în celelalte două funcții.

Se observă că, în cadrul fiecărei funcții, acțiunile efectuate sunt alese determinist. Astfel, la fiecare pas se alege o acțiune posibilă sau se întrerupe căutarea și se semnalează faptul că șirul nu poate fi obținut.

Funcțiile se apelează recursiv. Pentru a evita depășirea stivei, recursivitatea se poate simula iterativ.

Analiza complexității

La fiecare moment, deciziile se iau în timp constant.

Ordinul de complexitate al operației de citire a datelor de intrare este $O(L)$, unde L reprezintă suma lungimilor șirurilor de perle din fișierul de intrare.

Fiecare caracter din fiecare șir este parcurs o singură dată.

Ordinul de complexitate al algoritmului este $O(L_i)$ pentru un șir de perle de lungime L_i.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(L)$.

P030422: Romeo și Julieta

Problema se rezolvă folosind *algoritmul lui Lee*.

Se aplică acest algoritm folosind ca puncte de start poziția lui *Romeo* și poziția *Julietei*.

Vom folosi o matrice D în care vom pune valoarea 1 peste tot pe unde nu se poate trece, valoarea 2 în poziția în care se află *Romeo* inițial, valoarea 3 în poziția în care se află *Julieta* inițial și valoarea 0 în rest.

La fiecare pas k vom parcurge această matrice și în pozițiile vecine celor care au valoarea 2 vom pune valoarea 2, dacă acestea au valoarea 0 sau 3. Dacă o poziție are valoare 3, înseamnă că la un moment de timp anterior *Julieta* se putea afla în poziția respectivă. La același pas k vom mai parcurge matricea o dată și în pozițiile vecine celor care au valoarea 3 vom pune valoarea 3, dacă acestea au valoarea 0.

Dacă la pasul k poziția vecină uneia care are valoarea 3, are valoarea 2, atunci ne vom opri și k reprezintă momentul minim de timp după care ce-i doi se întâlnesc, iar poziția care are valoare 2 reprezintă locul întâlnirii.

La prima vedere s-ar părea că numărul k nu reprezintă momentul de timp minim la care cei doi se întâlnesc. Vom demonstra că algoritmul este corect prin metoda reducerii la absurd. Pentru aceasta avem în vedere că pozițiile marcate cu 2 reprezintă toate locurile în care se poate afla *Romeo* după cel mult k pași, iar cele marcate cu 2 reprezintă toate locurile în care se poate afla *Julieta* după cel mult k pași. Dacă k nu reprezintă momentul de timp minim la care cei doi se întâlnesc înseamnă că acesta a fost determinat mai devreme și algoritmul s-a oprit deja.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(M \cdot N)$.

Ordinul de complexitate al acestui algoritm este $O(k \cdot M \cdot N)$, unde k reprezintă momentul în care cei doi se întâlnesc.

Ordinul de complexitate al operației de scriere a rezultatului este $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(k \cdot M \cdot N)$.

Clasele a XI-a și a XII-a

P030423: Moșia lui Păcală

Această problemă se poate împărți în trei subprobleme mai mici.

Prima subproblemă constă în ordonarea punctelor în sens trigonometric sau anti-trigonometric pentru ca șirul lor să descrie un poligon.

Pentru aceasta vom alege cel mai de jos punct și îl vom pune pe prima poziție, iar apoi vom sorta celelalte puncte în funcție de unghiurile formate de segmentul care are ca și capete punctul ales la început și un punct dintre cele nesortate și dreapta orizontală care trece prin punctul ales la început.

După ce prima subproblemă a fost rezolvată se poate trece la rezolvarea celei de-a doua subprobleme.

A doua subproblemă constă în a determina surplusul de suprafață pe care îl aduce mutarea fiecăruia dintre pari și acesta este egal cu produsul dintre distanța maximă pe care poate fi mutat un par și distanța dintre cei doi pari vecini ai săi împărțită la 2.

La acest rezultat se ajunge prin câteva calcule simple din geometrie ținându-se cont de faptul că dacă mutăm un par, nu vom mai putea muta vecinii săi datorită restricțiilor impuse în enunț. Aceste surplusuri (costuri) sunt reținute într-un vector.

A treia subproblemă constă în a determina suprafața maximă cu care poate fi mărită moșia ținând cont de restricții.

Această subproblemă se rezolvă prin metoda programării dinamice și este asemănătoare problemei "*Oo*" care a fost propusă spre rezolvare la concursul de programare *Stelele Informaticii* al cărei enunț poate fi găsit în *GInfo* 8/2003, iar rezolvarea în *GInfo* 1/2004.

Diferența dintre aceste două probleme este dată de faptul că la problema *Oo*, fermierul *Ion* poate aduna ouăle din două sectoare și nu mai poate aduna ouăle din sectoarele vecine celor din care a adunat.

În cazul acestei subprobleme va trebui să construim un tablou unidimensional A cu N elemente. Elementul de pe o poziție i indică suprafața maximă cu care poate fi mărită moșia prin mutarea parului i.

Soluția este dată de cea mai mare valoare din acest vector.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(N)$.

Ordinul de complexitate a primei subprobleme este dat de ordinul de complexitate al operației de sortare utilizate și este $O(N \cdot \log N)$, dacă se folosește sortarea rapidă.

Ordinul de complexitatea a celei de-a doua și a treia subprobleme este $O(N)$.

Operația de scriere a rezultatelor se realizează în $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N \cdot \log N)$.

P030424: Lanterna

Se cere tipul de lanternă cu consumul cel mai mic, dintre cele cu care se poate ajunge la destinație în timp minim.

Cerința sugerează ideea folosirii căutării binare. Astfel, inițial se încearcă ajungerea la destinație cu $W_{min} = K$. Dacă se reușește ajungerea la destinație cu timpul minim, se va încerca cu un W_{min} mai mic; dacă nu se reușește, sau timpul obținut este mai mare decât timpul minim, se va încerca un W_{min} mai mare. Evident, timpul minim T_{min} este obținut pentru $W_{min} = K$.

Acestea fiind stabilite, avem de rezolvat următoarea subproblemă: *cum determinăm timpul minim în care putem ajunge la destinație pentru un anumit* W_{min}?

Construim o matrice A cu N linii și $W_{min} + 1$ coloane (de la 0 la W_{min}), în care $A_{i,j}$ semnifică timpul minim în care se poate ajunge din punctul de plecare în obiectivul i, având încă j *Wați* disponibili în lanternă pentru a continua mai departe.

Regulile de completare sunt următoarele:

- dacă i este un obiectiv prieten, de fiecare dată se va completa numai $A_{i,W_{min}}$, deoarece lanterna poate fi reîncărcată;
- dacă am completat $A_{i,j}$ cu o valoare V, atunci pentru orice drum de la i la un alt obiectiv p, de lungime L și consum C, am putea pune valoarea $V + L$ în $A_{p,j+C}$ dacă nu există deja o valoare mai mică. Evident, dacă $j + C > W_{min}$ această opțiune nu este luată în considerare. Dacă p este un obiectiv prieten, se folosește regula de mai sus.

Pentru a nu suprascrie prea des valori, vom completa elementele matricei A în ordinea timpului în care se poate ajunge în orașe. Pentru aceasta folosim o coadă de priorități, care va conține elemente de tipul (*obiectiv*, *waținRămași*, *timpDeAjungere*), sortată după *timpDeAjungere*. Se extrage repetat elementul cu timpul de ajungere minim și, folosind regulile prezentate, se adaugă noi elemente în coadă, dacă este cazul, completându-se și pozițiile corespunzătoare din matricea A.

Putem implementa coada de priorități cu ajutorul unui *heap*.

Dacă dorim să nu suprascriem deloc valori, combinăm acest algoritm cu algoritmul lui *Dijkstra*. Lăsăm această îmbunătățire ca exercițiu pentru cititor.

În final se vor afișa valorile cerute.

Analiza complexității

Se execută $\log K$ pași.

La fiecare pas se completează $O(W_{min} \cdot N)$ poziții; pentru fiecare completare a unei pozitii se execută $O(\log W_{min} \cdot N)$ operații. Pentru acest calcul, considerăm împreună operațiile de introducere, respectiv extragere ale unui anumit element în/din coada de priorități.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(\log K \cdot \log(W_{min} \cdot N) \cdot (W_{min} \cdot N))$.

În practică, nu toate pozițiile din matrice sunt completate, ceea ce face ca algoritmul să ruleze foarte repede.

Rundele #26 - #40

În cadrul acestui articol vom publica detalii despre desfășurarea rundelor #26-#40 ale ediției 2003/2004 a concursului de programare Bursele Agora, organizat de revista noastră.

P030425: Perechi

Există mai multe metode de rezolvare pentru această problemă. O variantă care duce la obținerea punctajului maxim este strategia *greedy* care va fi descrisă în continuare.

Pornind de la perechea (a, b) vom încerca să determinăm toate perechile (a', b') care pot fi obținute prin operații de scădere (prima operație permisă) și inversare (a treia operație permisă) numai dacă scăderea ar duce la obținerea unei valori negative.

La fiecare pas vom transforma perechea curentă (a', b') în perechea (b', a') dacă $(a' < b')$ sau în perechea $(a' - b', b')$ în caz contrar. Ne vom opri în momentul în care vom obține valoarea 0.

În continuare vom aplica aceiași pași pentru perechea (c, d). Vom păstra toate perechile (a', b') și (c', d') obținute.

Evident, pentru fiecare astfel de pereche este cunoscut numărul de pași necesari obținerii ei.

În cazul în care identificăm două perechi $(a", b")$ și $(c", d")$ pentru care $a" = c"$ și $b" = d"$ sau pentru care $a" = d"$ și $b" = c"$ vom ști că perechea (c, d) poate fi obținută prin transformări succesive pornind de la perechea (a, b). În cea de-a doua situație ($a" = d"$ și $b" = c"$) va trebui să efectuăm o inversare suplimentară.

Este ușor de observat că vom efectua scăderi și inversări până în momentul în care obținem perechea $(a", b")$, eventual efectuăm încă o inversare și apoi, vom efectua adunări și inversări până în momentul în care obținem perechea (c, d).

Evident, dintre toate perechile $(a", b")$ care satisfac proprietatea amintită, o vom alege pe cea pentru care numărul operațiilor efectuate este minim.

Au rezolvat corect...

Problema s-a dovedit a fi mult mai dificilă decât ar părea la prima vedere; doar trei dintre cei 32 de concurenți care au participat la această rundă au obținut punctajul maxim:

Mugurel-Ionuț Andreica, România
Marius Dumitran, România
Sorin Stancu-Mara, România

P030426: Triunghi

Folosind teorema lui *Pitagora* observăm imediat că va trebui să determinăm numărul perechilor (b, c) care satisfac următoarele condiții:

- $a^2 + b^2 = c^2$;
- $a < b < c$;
- b și c sunt numere întregi.

Pentru început vom transforma prima relație pentru a descoperi o modalitate mai simplă de determinare a perechilor (b, c).

Avem:

$$a^2 + b^2 = c^2$$
$$a^2 = c^2 - b^2$$
$$a^2 = (c + b) \cdot (c - b)$$

Observăm acum că pentru a obține soluții întregi este necesar (dar nu și suficient) ca numărul $c - b$ să fie divizor al numărului a^2.

Ca urmare, vom lua în considerare toți divizorii d ai numărului a^2 și apoi vom rezolva sistemul de ecuații:

$$c - b = d$$
$$c + b = a^2 / d$$

Rezolvând acest sistem de ecuații obținem valorile:

$$c = (a^2 / d + d) / 2$$
$$b = (a^2 / d - d) / 2$$

Acum vom verifica dacă aceste valori satisfac cele trei condiții amintite.

Principala dificultate o constituie determinarea valorilor d deoarece nu poate fi utilizat un algoritm simplu (valoarea a^2 poate fi foarte mare).

Vom determina o descompunere în factori primi a numărului a. Aceasta va avea forma:

$$a = d_1^{p_1} \cdot d_2^{p_2} \cdot \ldots \cdot d_k^{p_k}.$$

Evident, descompunerea în factori primi a numărului a^2 va avea forma:

$$a^2 = d_1^{2 \cdot p_1} \cdot d_2^{2 \cdot p_2} \cdot \ldots \cdot d_k^{2 \cdot p_k}.$$

Pe baza acestei descompuneri deducem că orice divizor al numărului a^2 va avea forma:

$$d = d_1^{q_1} \cdot d_2^{q_2} \cdot \ldots \cdot d_k^{q_k}.$$

unde:

- $0 \le q_1 \le 2 \cdot p_1$;
- $0 \le q_2 \le 2 \cdot p_2$;

...

- $0 \le q_k \le 2 \cdot p_k$.

Așadar, problema determinării divizorilor numărului a^2 se reduce la problema determinării produsului cartezian al mulțimilor $\{0, 1, ..., 2 \cdot p_1\}$, $\{0, 1, ..., 2 \cdot p_2\}$, ..., $\{0, 1, ..., 2 \cdot p_k\}$. Fiecărui element al produsului cartezian îi va corespunde un divizor. De exemplu, divizorul $d = d_1^{q_1} \cdot d_2^{q_2} \cdot \ldots \cdot d_k^{q_k}$ corespunde elementului $(q_1, q_2, ..., q_k)$.

Se observă că numărul mulțimilor pentru care este construit produsul cartezian este de cel mult nouă deoarece produsul primelor zece numere prime este mai mare decât 2.000.000.000 (valoarea maximă a numărului a):

$$2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 = 6.469.693.230.$$

De asemenea, se observă că suma maximă a puterilor este $2 \cdot 30 = 60$ deoarece avem $2^{31} = 2.147.483.648$.

Deducem că algoritmul de determinare a acestui produs cartezian (și a divizorilor numărului a^2) va rula suficient de repede pentru a se respecta limita de timp admisă.

O altă dificultate importantă este dată de precizia calculelor. Pentru a nu apărea erori din această cauză era necesară simularea operațiilor cu numere mari.

Au rezolvat aproape corect...

Dintre cei 34 de concurenți care au participat la această etapă a concursului, nici unul nu a reușit să obțină punctajul maxim. Cel mai mare punctaj a fost de 70 de puncte și a fost obținut de trei concurenți. Un singur alt concurent a reușit să obțină un punctaj de peste 50 de puncte.

Liviu Ciortea, România	*70*
Marius Dumitran, România	*70*
Alexandru Popa, România	*70*
Sorin Stancu-Mara, România	*51*

P030427: Fotbal

Vom lua în considerare toate submulțimile care se pot forma folosind primele $[n / 2]$ oi și le vom sorta în funcție de valoarea fotbalistică totală a oilor care fac parte dintr-o submulțime.

În continuare vom determina toate submulțimile formate din ultimele $n - [n / 2]$ oi. Pentru fiecare astfel de submulțime A, vom căuta în lista primelor submulțimi (folosind căutarea binară) acea submulțime B care are valoarea cea mai apropiată de $S / 2 - x$, unde S este valoarea fotbalistică însumată a tuturor oilor, iar x este valoarea fotbalistică totală a oilor din mulțimea A.

Echipa "candidată" va fi obținută prin reuniunea mulțimilor A și B. Dacă valoarea fotbalistică a echipei este mai apropiată de $S / 2$ decât cea mai bună soluție obținută anterior, atunci avem acum o nouă diferență minimă și o singură posibilitate (deocamdată) de a o obține. Dacă diferența este egală cu cea curentă, atunci vom crește numărul posibilităților.

În final vom scrie valoarea diferenței minime, precum și numărul posibilităților prin care aceasta poate fi obținută.

A rezolvat corect...

Un singur participant (dintre cei 43 care au trimis soluții la această problemă) a reușit să obțină punctajul maxim:

Marius Nicolae, România

P030428: Dispozitive de comunicare

Dispozitivele de comunicare pot fi considerate a fi nodurile unui graf orientate ale cărui arce descriu legăturile de comunicare (și direcțiile acestora).

În aceste condiții problema se reduce la determinarea numărului componentelor tare-conexe într-un graf orientat.

Au rezolvat corect...

Dintre cei 34 de participanți la această rundă a concursului de programare *Bursele Agora*, 14 au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Andrei Blaj, România
Liviu Ciortea, România
Claudiu-Dan Gheorghe, România
Marius Nicolae, România
Mircea Pașoi, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Alexandru Peptan, România
Vlad Petcu, România
Alexandru Popa, România
Claudiu Rad, România
Adrian Vladu, România

P030429: Media aritmetică

Se observă imediat că valoarea căutată este chiar media aritmetică a celor n numere date. Dacă notăm valoarea căutată prin m, atunci suma celorlalte $n - 1$ numere este $(n - 1) \cdot m$. Suma tuturor celor n numere va fi $(n - 1) \cdot m + m = n \cdot m$, iar media lor este $n \cdot m / n = m$.

După determinarea mediei, vor trebui căutate toate aparițiile sale în fișierul de intrare.

Principale dificultate o constituie faptul că numerele din fișierul de intrare nu pot fi memorate. Ca urmare, fișierul va trebui citit de două ori (o dată pentru determinarea mediei și o dată pentru determinarea aparițiilor). Din aceste motive, pentru citirea și scrierea datelor trebuie utilizate *buffer*-e de dimensiuni mari. În caz contrar, programele nu vor respecta limita de timp admisă.

Au rezolvat corect...

La această a treizecea etapă a concursului am primit 38 de rezolvări. Patru dintre cei care au participat la această rundă au obținut punctajul maxim:

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Mihail-Cosmin Pițˇ-Rada, România
Sorin Stancu-Mara, România

P030430: Keops

Problema se reduce la determinarea unui simplu drum într-un labirint, deci poate fi rezolvată folosind, de exemplu, algoritmul lui *Lee*.

Numărul minim de minute va fi dat de partea întreagă superioară a jumătății distanței minime dintre pozițiile inițiale ale celor doi hoți.

Au rezolvat corect...

Dintre cei 27 de concurenți care au trimis soluții la această problemă, 12 au obținut punctajul maxim:

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Liviu Ciortea, România
Adrian Diaconu, România
Kazmenko Ivan, Federația Rusă
Kiril Minkov, Bulgaria
Mihai Moalfă, România
Marius Nicolae, România
Csaba Pătcaș, România
Mihail-Cosmin Pițˇ-Rada, România
Alexandru Popa, România
Claudiu Rad, România

P030431: Colectare

Pozițiile obiectelor din labirint pot fi considerate a fi vârfurile unui graf neorientat. Cu ajutorul algoritmului lui *Lee* pot fi determinate costurile muchiilor dintre aceste noduri (costul unei muchii este dat de distanța minimă dintre obiectele corespunzătoare nodurilor care reprezintă extremitățile muchiei respective).

Vom adăuga un nod corespunzător poziției inițiale care va fi legat, în același mod, prin muchii (cu aceeași semnificație) de nodurile corespunzătoare obiectelor. În sfârșit, vom adăuga un ultim nod care va corespunde ieșirii din labirint. Acest nod va fi legat de toate celelalte noduri prin muchii ale căror lungime va fi dată de numărul minim de pași necesari pentru a ieși din labirint pornind din pozițiile corespunzătoare celorlalte noduri ale grafului.

În aceste condiții problema se reduce la determinarea unui drum hamiltonian de cost minim care pornește din nodul corespunzător poziției inițiale a lui *Merlin* și ajunge în nodul corespunzător ieșirii din labirint.

Se observă că graful construit are cel mult șapte noduri, deci nu apar probleme dacă folosim un algoritm exponențial pentru determinarea drumului hamiltonian.

Au rezolvat corect...

La această rundă au participat 38 de concurenți dintre care 18 au obținut punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Andrei Blaj, România
Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Kazmenko Ivan, Federația Rusă
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Mihai Moalfă, România
Mircea Pașoi, România
Daniel Păsăilă, România
Csaba Pătcaș, România
Mihail-Cosmin Pițˇ-Rada, România
Tudor-Alexandru Rabaea, România
Claudiu Rad, România
Andrei-Ionuț Tudor, România
Adrian Vladu, România

P030432: Lego

Aceasta este o problemă clasică pentru a cărei rezolvare trebuie aplicată metoda *backtracking*. Evident, un *backtracking* simplu (chiar folosind o mulțime de optimizări) nu ar fi dus la obținerea unui punctaj foarte mare fiindcă limita de timp ar fi fost depășită.

Puteau fi utilizate diferite tehnici euristice sau a unor algoritmi probabiliști cu ajutorul cărora s-ar fi obținut punctaje mai mari.

Au rezolvat aproape corect...

Dintre cei 30 de concurenți care au participat la această etapă a concursului, nici unul nu a reușit să obțină punctajul maxim. Cel mai mare punctaj a fost de 58 de puncte și a fost obținut de trei concurenți. Un singur alt concurent a reușit să obțină un punctaj de peste 50 de puncte.

Adina Pampu, România	*58*
Alexandru Peptan, România	*58*
Claudiu Rad, România	*58*
Mugurel-Ionuț Andreica, România	*57*

P030433: Cele două tipuri de turnuri

Pentru început vom determina teritoriile celor două specii. Acestea sunt date de înfășurătorile convexe ale punctelor care reprezintă turnurile. După determinarea acestor înfășurători putem verifica foarte simplu, pentru fiecare turn în parte, dacă se află în interiorul teritoriului celeilalte specii (vom utiliza un algoritm de verificare a faptului dacă un punct se află în interiorul unui poligon convex).

Au rezolvat corect...

Dintre cei 25 de participanți la această etapă, opt au reușit să obțină numărul maxim de puncte.

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Andrei Cârcu, România
Vasanth Jai, India
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Csaba Pătcaș, România
Sorin Stancu-Mara, România

P030434: 008 - James Bomb

De această dată avem de a face cu un labirint tridimensional. Cu toate acestea algoritmul aplicat este același (*Lee*).

Vom determina distanța până la bombă și, în cazul în care *James Bomb* poate ajunge lângă bombă înainte ca aceasta să explodeze, vom scrie rezultatul în fișierul de ieșire.

În caz contrar, vom determina distanța minimă din poziția inițială până la marginea clădirii.

Au rezolvat corect...

Problema s-a dovedit a fi relativ simplă; 22 dintre cei 38 de concurenți care au participat la această rundă au obținut punctajul maxim:

Adrian Diaconu, România
Silviu-Ionut Ganceanu, România
Sergiu Hlihor, România
Vasanth Jai, India
Andrei Janca, România
Stelian Kelemen, România
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Mihai Moalfă, România
Mircea Pașoi, România
Daniel Păsăilă, România
Mihail-Cosmin Pițt-Rada, România
Claudiu Rad, România
Paul-Horațiu Stan, România
Valentin Stanciu, România
Sorin Stancu-Mara, România
Andrei-Ionuț Tudor, România
Adrian Vladu, România

P030435: Raze LASER

Pentru fiecare particulă radioactivă vom calcula tangenta unghiului format de dreapta care o unește de origine cu axa orizontală.

Vom alege apoi particulele pentru care obținem cea mai mică și cea mai mare valoare a tangentei și vom determina unghiul format de dreptele care unesc cele două particule de origine.

Trebuie observat faptul că în locul tangentei se poate utiliza orice altă funcție f care păstrează ordinea (în cazul în care $a < b$, avem întotdeauna fie $f(a) < f(b)$, fie $f(b) < f(a)$). Așadar, se pot utiliza funcțiile sinus sau cosinus, dar și altele care satisfac proprietatea amintită.

Au rezolvat corect...

Problema a fost una dintre cele mai simple probleme propuse spre rezolvare la această ediție a concursului nostru; aproape trei sferturi (23) dintre cei 31 de concurenți care au participat la această rundă au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Cristian Cernat, România
Liviu Ciortea, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionut Ganceanu, România
Claudiu-Dan Gheorghe, România
Sergiu Hlihor, România
Andrei Janca, România
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Adina Pampu, România
Ciprian Păduraru, România
Csaba Pătcaș, România
Alexandru Peptan, România
Mihail-Cosmin Pițt-Rada, România
Bogdan-Ștefan Popescu, România
Tudor-Alexandru Rabaea, România
Claudiu Rad, România
Sorin Stancu-Mara, România
Adrian Vladu, România

P030436: Sertare

Se observă imediat că putem considera o bilă neagră ca fiind o paranteză deschisă și o bilă albă ca fiind o paranteză închisă. În aceste condiții problema se reduce la determinarea numărului de parantezări corecte pentru n perechi de paranteze. Această valoare este dată de cunoscutul număr al lui *Catalan*.

Au rezolvat corect...

Problema era foarte cunoscută, motiv pentru care a fost rezolvată corect de mulți participanți. Dintre cei 28 de con-

curenți care au trimis soluții, 16 au reușit să obțină punctajul maxim.

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Andrei Blaj, România
Adrian Cârcu, România
Liviu Ciortea, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionut Ganceanu, România
Sergiu Hlihor, România
Vasanth Jai, India
Haryanto Lego, Statele Unite ale Americii
Mihai Moalfă, România
Mircea Pașoi, România
Daniel Păsăilă, România
Mihail-Cosmin Pițt-Rada, România
Tudor-Alexandru Rabaea, România

P030437: Rețea de calculatoare

Problema se reduce la determinarea valorii unei tăieturi minime într-un graf neorientat. Deoarece potrivit teoremei *flux maxim - tăietură minimă* această valoare este egală cu cea a fluxului maxim, putem aplica orice algoritm de determinare a acestuia.

Au rezolvat corect...

Și această problemă este foarte cunoscută și a fost rezolvată corect de o mare parte dintre cei care au trimis soluții. Dintre cei 28 de concurenți, 19 au reușit să obțină punctajul maxim:

Mugurel-Ionuț Andreica, România
Andrei Blaj, România
Adrian Cârcu, România
Cristian Cernat, România
Liviu Ciortea, România
Marius Dumitran, România
Silviu-Ionut Ganceanu, România
Andrei Gönczi, România
Stelian Kelemen, România
Haryanto Lego, Statele Unite ale Americii
Kiril Minkov, Bulgaria
Mircea Pașoi, România
Ciprian Păduraru, România
Daniel Păsăilă, România
Alexandru Popa, România
Bogdan-Ștefan Popescu, România
Claudiu Rad, România
Sorin Stancu-Mara, România
Alexandru-Cosmin Vlăjoagă, România

P030438: Extratereștrii

Deși se observă imediat că problema cere determinarea celui mai lung subșir comun a două șiruri de caractere, apar dificultăți datorate faptului că nu putem folosi versiunea clasică de rezolvare bazată pe metoda programării dinamice. Ca urmare, pentru a rezolva problema trebuia utilizat algoritmul de determinare a celui mai lung subșir comun folosind un spațiu de memorie liniar.

Au rezolvat corect...

Dintre cei 35 de participanți la această rundă a concursului de programare *Bursele Agora*, zece au reușit să obțină punctajul maxim:

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Andrei Blaj, România
Andrei Cârcu, România
Adrian Diaconu, România
Vasanth Jai, India
Haryanto Lego, Bulgaria
Daniel Păsăilă, România
Mihail-Cosmin Pițt-Rada, România
Adrian Vladu, România

P030439: Matrice magică

O posibilitate de rezolvare a acestei probleme constă în construirea a două matrice X și Y pe baza matricei A. Valorile x_{ij} vor reprezenta cele mai mari numere divizibile cu p care sunt mai mici decât a_{ij}, iar valorile y_{ij} vor reprezenta cele mai mari numere divizibile cu p care sunt mai mici decât a_{ij}. Vom numi matricele X și Y ***matrice inferioară***, respectiv ***matrice superioară***.

Matricea inferioară va fi bordată cu sumele pe linii și pe coloane ale matricei A, iar *matricea superioară* va fi bordată cu sumele pe linii și coloane ale matricei Y. Elementele celor două matrice se împart la p și se calculează diferența lor. Se obține astfel o matrice binară care este bordată de anumite sume. Astfel problema se reduce la determinarea muchiilor unui graf pentru care se cunosc gradele nodurilor. Așadar, am obținut o problemă care se rezolvă la fel ca problema *Pluto* propusă spre rezolvare la runda #05 a acestei ediții a concursului.

Au rezolvat corect...

La această a patruzecea etapă a concursului am primit 25 de rezolvări. Zece dintre cei care au participat la această rundă au obținut punctajul maxim:

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Adrian Diaconu, România
Marius Dumitran, România
Silviu-Ionut Ganceanu, România
Kiril Minkov, Bulgaria
Mircea Pașoi, România
Csaba Pătcaș, România
Alexandru Popa, România
Adrian Vladu, România

Bursele Agora 2003/2004 Top 50

1. Mugurel-Ionuț Andreica, București	3524
2. Adrian Cârcu, Bistrița	3205
3. Mircea Pașoi, Ploiești	2928
4. Liviu Ciortea, București	2867
5. Haryanto Lego, Statele Unite ale Americii	2800
6. Kiril Minkov, Bulgaria	2756
7. Silviu-Ionuț Gănceanu, Bârlad	2671
8. Chattopadhyay Amit, India	2634
9. Marius Dumitran, București	2589
10. Daniel Păsăilă, Suceava	2562

11. M.-Cosmin Pițˇ-Rada, Dr. Tr. Severin	2451	31. Ciprian Păduraru, Bârlad	868
12. Vasanth Jai, India	2411	32. Valentin Stanciu, București	822
13. Csaba Pătcaș, Oradea	2387	33. Claudiu-Dan Gheorghe, Slobozia	798
14. Adrian Vladu, București	2325	34. Victor Podeanu, București	750
15. Adrian Diaconu, București	2200	35. Andrei-Ionuț Tudor, Slobozia	712
16. Claudiu Rad, Deva	2041	36. Eduard Băzăvan, Dr. Tr. Severin	709
17. Ivan Kazmenko, Federația Rusă	1966	37. Ioan-Dan Docleanu, Nădlac	686
18. Alexandru Peptan, Dr. Tr. Severin	1800	38. Stelian Kelemen, Bistrița	679
19. Alexandru Popa, București	1723	39. Dalian-Valeriu Moraru, București	617
20. Sorin Stancu-Mara, București	1711	40. Adina Pampu, Făgăraș	612
21. Andrei Blaj, Suceava	1547	41. Irinel Besliu, Vaslui	592
22. Tudor-Alexandru Rabaea, Constanța	1467	42. Sorin Vultureanu	567
23. Sergiu Hlihor, Lugoj	1446	43. Adrian-Romulus Ciripan, Alba-Iulia	540
24. Mihai-Vlad Pantiș, Cluj-Napoca	1267	44. Dragoș Cavulea, Botoșani	539
25. Mihai Moalfă, Sîngeorz-Băi	1210	45. Călin Bindea, Bistrița	497
26. Cristian Cernat, Târgu Jiu	1124	46. Vlad Petcu, Boldești-Scăieni	494
27. Andrei Gönczi, Arad	1120	47. Renald, Indonezia	460
28. Andrei Janca, Dr. Tr. Severin	1097	48. Cosmin Molea, Bascov	450
29. Marius Nicolae, Slatina	1045	49. Johannes Slotta, Germania	396
30. Iulian Blajuț, Luncani	869	50. Ștefan Gheorghe, București	383

Rundele #41 - #50

În continuare vă vom prezenta enunțurile celor 10 probleme propuse spre rezolvare la semifinala ediției din acest an a concursului de programare Bursele Agora care a avut loc pe data de 21 martie.

P040429: Minele de aur

Elmer Fud și *Porky Pig* sunt prieteni de foarte mult timp și au decis să meargă împreună în căutare de aur.

Ei au reușit să sape mai multe mine din care pot extrage aur în cantități foarte mari și s-au hotărât să se asocieze și să intre în afaceri cu aur. La un moment dat, când treburile mergeau bine, ei s-au certat și au decis să împartă minele și afacerile în două. A fost ușor să împartă afacerile, dar împărțitul minelor a devenit o problemă serioasă deoarece doresc să construiască un gard liniar și de fiecare parte a gardului să se găsească un număr egal de mine.

Elmer și *Porky* apelează la ajutorul vostru pentru a le spune unde să construiască gardul.

Date de intrare

Fișierul de intrare **GOLDMINE.IN** conține pe prima linie un singur număr n, care reprezintă numărul de mine care trebuie să revină fiecăruia dintre cei doi.

Fiecare a i-a linie dintre următoarele $2 \cdot n$ linii conține câte 2 numere, separate între ele printr-un singur spațiu, care reprezintă coordonatele unei mine.

Date de ieșire

Fișierul de ieșire **GOLDMINE.OUT** trebuie să conțină pe o singură linie trei numere a, b și c separate între ele prin spații, care reprezintă coeficienții dreptei de ecuație $a \cdot x + b \cdot y + c = 0$ care are de o parte sau pe ea n mine și de cealaltă parte sau pe ea celelalte n mine.

Restricții și precizări

- $0 \le n \le 30.000$;
- coeficienții a, b și c vor fi scriși cu opt zecimale exacte;
- pot exista două sau mai multe mine aflate la aceleași coordonate;
- coordonatele minelor sunt numere întregi cuprinse între 1 și 1000.

Exemplu

GOLDMINE.IN

```
2
1 2
2 2
3 1
3 2
```

GOLDMINE.OUT

```
3.00000000 -2.00000000 -3.00000000
```

Timp maxim de execuție/test: 0,5 secunde

P040430: Rivali

Două companii distribuie reviste într-un județ în care există un număr de n orașe și m șosele pe care se poate circula în ambele sensuri. Pentru fiecare șosea se cunoaște costul unui drum pe șoseaua respectivă. Fiecare companie dorește să pună la punct o rețea de distribuție care să-i permită să poată transporta reviste dintr-un oraș în oricare altul. Evident, fiecare companie va dori să obțină o rețea de distribuție al cărei cost total (suma costurilor șoselelor din rețea) să fie minim. Din nefericire, cele două companii rivale nu își pot permite să aibă rețele de distribuție identice. Ca urmare, ele vor alege rețele de distribuție puțin diferite. Va trebui să determinați diferența dintre costurile totale ale celor două rețele de distribuție știind că aceste două costuri sunt cele mai mici posibile și că rețelele nu pot fi complet identice.

Date de intrare

Prima linie a fișierului de intrare **RIVALS.IN** conține numărul n al orașelor și numărul m al șoselelor. Fiecare dintre următoarele m linii conține trei numere întregi x, y și c cu semnificația: *există o șosea între orașele x și y, al cărei cost este c.* Numerele de pe o linie vor fi separate prin spații.

Date de ieșire

Fișierul de ieșire **RIVALS.OUT** trebuie să conțină o singură linie pe care se va afla diferența dintre costurile totale ale celor două rețele de distribuție.

Restricții și precizări

- $2 \le n \le 500$, $1 \le m \le 5000$;
- orașele sunt identificate prin numere cuprinse între 1 și n;
- există posibilitatea ca diferența să fie 0 dacă cele două rețele de distribuție sunt diferite, dar au același cost total;

- costurile șoselelor sunt numere întregi cuprinse între 1 și 1000.

Exemplu

RIVALS.IN	RIVALS.OUT
4 5 1 2 3 1 3 5 2 3 3 2 4 5 3 4 1	2

Timp maxim de execuție/test: 2 secunde

P040431: Cerc

Pluto se plictisea și a început să se joace. El are o foaie de hârtie cu pătrățele pe care o stropește cu cerneală și după aceea își notează pe o altă hârtie coordonatele centrelor petelor. *Pluto* dorește să traseze un cerc pe foaia pătată astfel încât în interiorul cercului să fie cel mult k dintre punctele ale căror coordonate le-a notat, iar în interior și pe frontieră (circumferință) să existe un număr de puncte mai mare sau egal cu k. Deoarece pe hârtie sunt foarte multe pete și *Pluto* dorește să rezolve astăzi problema determinării cercului, vă roagă să scrieți un program care, pe baza coordonatelor centrelor petelor să determine coordonatele centrului unui asemenea cerc și raza acestuia.

Date de intrare

Fișierul de intrare **CIRCLE.IN** conține pe prima linie două numere întregi n și k, separate între ele printr-un singur spațiu, care reprezintă numărul de pete, respectiv numărul de centre ale petelor care trebuie să fie luat în considerare la găsirea cercului. Pe fiecare dintre următoarele n linii se află câte două numere, separate printr-un singur spațiu, care reprezintă coordonatele centrului unei pete.

Date de ieșire

Fișierul de ieșire **CIRCLE.OUT** trebuie să conțină trei numere x, y și r, separate prin spații, care reprezintă coordonatele centrului și raza cercului care are proprietatea din enunț.

Restricții și precizări

- $1 \le k \le n \le 100.000$;
- coordonatele centrelor petelor sunt numere întregi cuprinse între 1 și 10.000;
- numerele din fișierul de ieșire vor fi scrise cu opt zecimale exacte.

Exemplu

CIRCLE.IN

```
4 3
0 10
20 10
10 5
10 15
```

CIRCLE.OUT

```
10.00000000 10.00000000 10.00000000
```

Timp maxim de execuție/test: 1 secundă

P040432: Parola

Într-un seif se află niște documente pe care trebuie să le extrageți. Problema este că seiful este prevăzut cu un terminal care necesită introducerea unei parole pentru a putea deschide seiful. La accesarea seifului, pe ecranul terminalului este afișat un cuvânt cheie format din litere mici ale alfabetului englezesc.

Parola este dată de cea mai mică rotație la stânga (în ordine lexicografică) a cuvântului cheie.

Date de intrare

Fișierul de intrare **PASSWORD.IN** conține pe prima linie un șir de caractere format din litere mici ale alfabetului englezesc.

Date de ieșire

Fișierul de ieșire **PASSWORD.OUT** trebuie să conțină un singur număr care reprezintă numărul de deplasări circulare la stânga ale șirului din fișierul de intrare necesare pentru a obține parola de acces cerută.

Restricții și precizări

- lungimea șirului din fișierul de intrare este un număr întreg cuprins între 1 și 100.000;
- dacă există mai multe soluții va fi aleasă cea care necesită un număr minim de deplasări circulare la stânga.

Exemplu

PASSWORD.IN	PASSWORD.OUT
mississippi	10

Timp maxim de execuție/test: 1 secundă

P040433: Coridorul

Se consideră un coridor dreptunghiular de dimensiuni $m \cdot n$. În acest coridor există mai multe coloane drepte de dimensiune neglijabilă.

Se cere să se determine raza celei mai mari mingi sferice care poate străbate coridorul din capătul de vest până în capătul de est. Pentru minge se poate alege orice punct de pornire aflat în capătul din vest și orice punct final aflat în capătul de est. Înălțimea coridorului este întotdeauna suficient de mare.

Date de intrare

Fișierul de intrare **HALLWAY.IN** conține două numere întregi m și n, separate între ele printr-un singur spațiu, care reprezintă dimensiunile coridorului pe direcțiile *est-vest*, respectiv *nord-sud*. Cea de-a doua linie conține un singur număr k, reprezentând numărul de coloane care se află în acest coridor.

Fiecare dintre următoarele k linii conține două numere, separate între ele printr-un singur spațiu, care reprezintă coordonatele la care se află o coloană.

Date de ieșire

Fișierul de ieșire **HALLWAY.OUT** trebuie să conțină un singur număr care reprezintă raza celei mai mari mingi sferice care poate străbate coridorul din capătul de vest până în capătul de est.

Restricții și precizări

- $1 \le m, n, k \le 1.000$;
- direcția *est-vest* este reprezentată de prima coordonată;
- raza determinată a mingii trebuie scrisă cu opt zecimale exacte.

Exemplu

HALLWAY.IN	HALLWAY.OUT
5 2 1 1 1	0.50000000

Timp maxim de execuție/test: 1 secundă

P040434: Ordinea

Se consideră n copii așezați în cerc și numerotați de la 1 la n în sens trigonometric. Copiii joacă următorul joc: jocul începe de la primul copil (cel al cărui număr de ordine este 1); la fiecare al i-lea pas al jocului se numără i copii în sens trigonometric și este eliminat copilul la care se ajunge; la pasul următor numărătoarea începe de la copilul care urmează după cel eliminat.

Așadar, dacă numărul copiilor este suficient de mare, la primul pas este eliminat al doilea copil, la al doilea pas al patrulea, la al treilea pas al șaptelea, apoi la al patrulea pas al unsprezecelea și așa mai departe. Va trebui să determinați ordinea în care vor fi eliminați copiii.

Date de intrare

Fișierul de intrare **ORDER.IN** conține pe prima linie un număr întreg n, care reprezintă numărul de copii.

Date de ieșire

Fișierul de ieșire **ORDER.OUT** trebuie să conțină o singură linie pe care se vor afla n numere distincte cuprinse între 1 și n care reprezintă numerele de ordine ale copiilor în ordinea în care au fost eliminați.

Restricție

- $2 \le n \le 30.000$.

Exemplu

ORDER.IN	ORDER.OUT
6	2 4 1 3 5 6

Timp maxim de execuție/test: 1 secundă

P040435: Pioni

Se consideră o tablă împărțită în mai multe celule care conțin pioni. Celulele sunt conectate între ele prin săgeți, și mergând de la orice celulă în direcția în care indică săgețile nu vom ajunge niciodată în locul din care am plecat. Deci, există celule din care nu pleacă săgeți și celule în care nu ajung săgeți.

Considerăm doi jucători. Fiecare jucător poate muta, când îi vine rândul, un pion din celula în care se află într-o celulă spre care pleacă o săgeată.

Cel care nu mai poate muta atunci când îi vine rândul pierde partida.

La începutul unui joc, prima mutare o face jucătorul 1.

Să se determine dacă jucătorul 1 are strategie sigură de câștig.

Date de intrare

Fișierul de intrare **PAWNS.IN** conține pe prima linie două numere întregi n și m, separate printr-un singur spațiu, care reprezintă numărul de celule de pe tablă, respectiv numărul de săgeți.

Pe fiecare dintre următoarele m linii se află câte două numere întregi x și y, separate între ele printr-un singur spațiu, cu semnificația că *există o săgeată care pleacă din celula cu numărul de ordine x și ajunge în celula cu numărul de ordine y.*

Pe linia următoare se află un număr t care reprezintă numărul de partide care se joacă.

Pe fiecare dintre următoarele t linii se află n numere întregi, separate între ele prin spații, care reprezintă numărul de pioni din fiecare celulă pentru o anumită partidă.

Date de ieșire

Fișierul de ieșire **PAWNS.OUT** trebuie să conțină t linii. Fiecare dintre cele t linii va conține valoarea 0, dacă pentru o configurație jucătorul 1 nu are strategie sigură de câștig și valoarea 1 în caz contrar.

Restricții și precizări

- $1 \le n, m \le 500$;
- $1 \le t \le 15$;
- numărul de pioni dintr-o celulă nu va depăși valoarea 500;
- celulele sunt numerotate de la 1 la n.

Exemplu

PAWNS.IN	PAWNS.OUT
3 3 1 2 1 3 2 3 2 1 0 0 2 2 2	1 0

Timp maxim de execuție/test: 1 secundă

P040436: Matrice binară

Se consideră o matrice binară de dimensiune $m \cdot n$ (elementele matricei sunt 0 sau 1).

Se cere să se determine aria maximă care poate fi acoperită de două dreptunghiuri care conțin numai elemente cu valoarea 0.

Date de intrare

Fișierul de intrare **BMATRIX.IN** conține pe prima linie două numere întregi m și n, separate printr-un singur spațiu, care reprezintă dimensiunile matricei.

Pe fiecare dintre următoarele m linii se află n numere care pot avea valorile 0 sau 1 și care nu sunt separate între ele prin spații.

Date de ieșire

Fișierul de ieșire **BMATRIX.OUT** trebuie să conțină un singur număr, care reprezintă aria maximă care poate fi acoperită de două dreptunghiuri care conțin numai elemente cu valoarea 0.

Restricții și precizări

- $1 \le m, n \le 200$;
- cele două dreptunghiuri nu se pot suprapune.

Exemplu

BMATRIX.IN	BMATRIX.OUT
6 8 10000000 10000000 11100011 00100011 00100011 00111111	23

Timp maxim de execuție/test: 1 secundă

P040437: Numere

Se consideră șirul obținut prin concatenarea numerelor naturale (în ordinea lor firească). Acest șir va avea forma:

123456789101112...

Va trebui să determinați poziția primei apariții în șirul considerat a unui număr k dat.

Date de intrare

Fișierul de intrare **NUMBERS.IN** conține pe prima linie un număr întreg n, care reprezintă numărul de numere care trebuie căutate în șirul dat. Pe fiecare dintre următoarele n linii se află câte un număr care trebuie căutat.

Date de ieșire

Fișierul de ieșire **NUMBERS.OUT** trebuie să conțină n linii. Pe fiecare dintre cele n linii trebuie să se afle un număr care reprezintă poziția primei apariții în șirul considerat a numărului corespunzător din fișierul de intrare.

Restricții și precizări

- $1 \le n \le 20$;
- numerele din fișierul de intrare sunt întregi și sunt cuprinse între 1 și 1.000.000.000;

Exemplu

NUMBERS.IN	NUMBERS.OUT
2 67 345	6 3

Timp maxim de execuție/test: 1 secundă

P040438: Cercuri

Se consideră zece cercuri pentru care se cunosc coordonatele centrelor si razele lor.

Să se determine cel mai mic pătrat care conține în interior cele zece cercuri.

Date de intrare

Fișierul de intrare **CIRCLES.IN** conține zece linii. Pe fiecare dintre acestea se află câte trei numere x, y și r, separate între ele prin spații, care reprezintă coordonatele centrului și raza unui cerc.

Date de ieșire

Fișierul de ieșire **CIRCLES.OUT** trebuie să conțină o singură linie pe care se va afla latura minimă a unui pătrat care conține în interior cele zece cercuri.

Restricții și precizări

- coordonatele centrelor cercurilor sunt numere întregi cuprinse între 1 și 999;
- razele cercurilor sunt numere întregi cuprinse între 1 și 100;
- coordonatele tuturor punctelor aflate pe circumferințele cercurilor sunt cuprinse între 0 și 1000;
- laturile pătratului nu sunt neapărat paralele cu axele de coordonate;
- valoarea din fișierul de ieșire va fi scrisă cu opt zecimale exacte.

Exemplu

CIRCLES.IN	CIRCLES.OUT
1 1 1 3 3 1 6 2 2 10 2 2 2 6 2 6 6 2 10 6 2 2 10 2 6 10 2 10 10 2	12.00000000

Timp maxim de execuție/test: 1 secundă

Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

În acest articol vom prezenta o clasă creată în redacția GInfo și modul în care se pot aplica conceptele de introspecție și reflexie în limbajul PHP.

Introspecție și reflexie în PHP

Introspecția reprezintă acțiunea prin care se pot afla informații despre metodele și proprietățile unei clase.

Reflexia reprezintă acțiunea prin care se pot introduce în mediul de execuție și în domeniul de vizibilitate al mediului de execuție curent – clase stocate în altă parte sau clase create de utilizator pentru a fi folosite cu anumite module care au fost realizate anterior de utilizator sau de alți programatori.

Termenii de *introspecție* și *reflexie* în practică nu pot fi separați, deoarece pentru a introduce un obiect în mediul de execuție, trebuie cunoscut numele, metodele și proprietățile acestuia și nu pot fi aflate informații despre o clasă care nu este încărcată și nu este vizibilă în mediul de execuție.

În continuare vom prezenta un set de funcții cu ajutorul cărora se pot afla informații despre clasele încărcate și vom exemplifica utilizarea acestora în cadrul unei clase similare clasei *Class* din limbajul *Java*.

Funcția get_declared_classes

Această funcție returnează un tablou care conține numele tuturor claselor definite în momentul execuției unui *script PHP*.

Funcția class_exists

Această funcție primește ca parametru un șir de caractere și returnează valoarea logică **TRUE** dacă există o clasă care să aibă numele identic cu șirul de caractere primit ca parametru și valoarea logică **FALSE** în caz contrar.

Funcția get_parent_class

Această funcție are un parametru care poate fi de tip șir de caractere sau obiect.

În cazul în care parametrul este de tipul șir de caractere, atunci funcția `get_parent_class` returnează un șir de caractere care reprezintă numele clasei care este părintele direct al clasei al cărui nume este dat de parametru.

În cazul în care parametrul este de tip obiect, atunci funcția returnează un șir de caractere care reprezintă numele părintelui direct al clasei a cărei instanță este reprezentată de obiectul dat ca parametru.

Funcția is_a

Această funcție are doi parametri. Primul parametru este de tip obiect, iar al doilea este de tip șir de caractere.

Funcția `is_a` returnează valoarea logică **TRUE** dacă primul parametru este de tipul reprezentat de al doilea parametru sau de un tip derivat din cel de-al doilea parametru și valoarea logică **FALSE** în caz contrar.

De exemplu, dacă avem clasele `cls1`, `cls2` și `cls3` și `cls3` este derivată din `cls2` și `cls2` este derivată din `cls1`, atunci, dacă `obj` reprezintă o instanță a clasei `cls3`, în urma apelurilor `is_a(obj, "cls1")`, `is_a (obj, "cls2")` și `is_a(obj, "cls3")` se obține valoarea logică **TRUE**.

Funcția is_subclass_of

Funcția `is_subclass_of` are aceiași parametri cu funcția anterioară. Funcționalitatea ei diferă față de funcția precedentă prin faptul că valoarea logică **TRUE** este returnată numai în cazul în care primul parametru este o instanță a unui tip derivat din tipul al cărui nume este dat de cel de-al doilea parametru.

În condițiile exemplului anterior, în urma apelului `is_subclass_of(obj, "cls3")` se va obține valoarea logică **FALSE**.

Funcția get_class_methods

Această funcție are un parametru care poate fi de tip șir de caractere care reprezintă numele unei clase sau de tip obiect.

Funcția `get_class_methods` returnează un tablou ale cărui elemente sunt de tipul șir de caractere și care repre-

zintă numele metodelor definite în cadrul clasei cu numele primit ca parametru sau în cadrul clasei care este reprezentată de obiectul primit ca parametru.

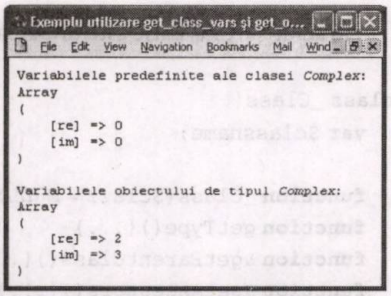


Figura 1: Exemplu de utilizare a funcţiilor get_class_vars şi get_object_vars

În cazul în care parametrul este de tip şir de caractere şi nu este definită nici o clasă cu acest nume, atunci funcţia returnează valoarea logică **FALSE**.

Funcţia get_class_vars

Această funcţie are un singur parametru de tip şir de caractere care reprezintă numele unei clase şi returnează un tablou ale cărui elemente sunt de tip şir de caractere şi care reprezintă valorile implicite ale variabilelor definite în cadrul clasei.

Tabloul rezultat în urma apelului acestei funcţii este indexat după numele variabilelor definite în cadrul clasei.

În continuare vă prezentăm un exemplu de utilizare a acestei clase, iar în figura 1 se poate observa efectul execuţiei *script*-ului:

```
<?PHP
  class Complex{
    var $re=0;
    var $im=0;

    function Complex($re = 0, $im = 0){
      $this->re = $re;
      $this->im = $im;
    }
    function add($a){
      return new Complex($a->re + $this->re,
                         $a->im + $this->im);
    }
    function toString(){
      if ($this->im==0)
        return (string) $this->re;
      if ($this->re==0)
        return $this->im.'i';
      if ($this->im<0)
        return $this->re.$this->im.'i';
      return $this->re.'+'.$this->im.'i';
    }
  }
?>

<HTML>
  <HEAD>
    <TITLE>
      Exemplu utilizare get_class_vars şi
                                get_object_vars
    </TITLE>
  </HEAD>
  <BODY>
    <PRE>
    <?PHP
      echo "Variabilele predefinite ale clasei
                              <I>Complex</I>:\n";
      print_r(get_class_vars('Complex'));
      echo "\nVariabilele obiec-
                                  tului de tipul
                              <I>Complex</I>:\n";
      $a = new Complex(2, 3);
      print_r(get_object_vars($a));
    ?>
    </PRE>
  </BODY>
</HTML>
```

Funcţia get_class

Această funcţie primeşte ca parametru un obiect şi returnează numele clasei care reprezintă tipul obiectului primit ca parametru.

Funcţia get_object_vars

Această funcţie are un singur parametru de tip obiect şi returnează un tablou ale cărui elemente sunt de tip şir de caractere şi care reprezintă valorile variabilelor definite în cadrul acestuia.

Tabloul rezultat în urma apelului acestei funcţii este indexat după numele variabilelor definite în cadrul obiectului.

În *script*-ul al cărui rezultat se poate observa în figura 1 am utilizat şi această funcţie.

Se poate observa că funcţia `get_object_vars` se aseamănă foarte mult cu funcţia `get_class_vars`.

Funcţia call_user_method

Această funcţie realizează apelul unei metode care aparţine unei clase sau unui obiect cu parametrii specificaţi.

Funcţia `call_user_method` are mai mulţi parametri. Primul parametru este de tip şir de caractere şi reprezintă numele metodei care se va apela. Al doilea parametru este de tip obiect sau de tip şir de caractere şi reprezintă numele unei clase. În cazul în care acest parametru este de tip obiect, metoda dată de primul parametru este apelată pentru obiect, iar dacă parametrul este de tip şir de caractere, atunci metoda este apelată pentru clasa al cărui nume este dat de acest parametru. Restul parametrilor reprezintă parametrii cu care se va apela metoda al cărui nume este dat de primul parametru.

Funcţia call_user_method_array

Această funcţie are aceeaşi funcţionalitate cu funcţia anterioară cu diferenţa că are doar trei parametri, iar al treilea

parametru este un tablou unidimensional care conține parametrii cu care se va apela metoda dată de primul parametru pentru clasa sau obiectul reprezentat de cel de-al doilea parametru.

Aceste ultime două funcții pot fi asociate reflexiei. Alături de acestea, pentru reflexie se mai pot utiliza funcțiile `include`, `include_once`, `require`, `require_once` care au fost prezentate în articolele anterioare și funcția `eval`.

Pentru introspecție, în afară de funcțiile prezentate în acest articol mai pot fi utilizate și funcțiile `dl`, `extension_loaded` și `get_declared_classes`.

În continuare vom prezenta funcțiile `eval`, `dl`, `get_declared_classes` și `extension_loaded`.

Funcția eval

Această funcție primește ca parametru un șir de caractere care reprezintă o secvență de cod *PHP* și realizează execuția acestuia.

Funcția dl

Această funcție realizează încărcarea în timpul execuției *script*-ului a unor extensii externe, dacă acest lucru este permis de setările interpretorului *PHP*.

Funcția `dl` primește ca parametru un șir de caractere care reprezintă calea către extensia care trebuie încărcată și returnează valoarea logică **`TRUE`** dacă s-a reușit încărcarea extensiei și valoarea logică **`FALSE`** în caz contrar.

Funcția extension_loaded

Această funcție verifică dacă o anumită extensie este încărcată. Funcția primește un parametru de tip șir de caractere care reprezintă numele unei extensii și returnează o valoare logică corespunzătoare.

Funcția get_declared_classes

Funcția `get_declared_classes` nu are nici un parametru și returnează un tablou unidimensional care conține numele tuturor claselor încărcate în mediul de execuție în momentul apelării acesteia.

Clasa _Class

În continuare vă prezentăm o clasă pe care am construit-o în vederea descoperirii de informații legate de clase prezente în mediul de execuție. Această clasă este experimentală și acoperă doar o parte a introspecției care poate fi realizată în cadrul limbajului *PHP*.

Metodele clasei `_Class` se bazează pe funcțiile prezentate la începutul acestui articol.

La realizarea acestei clase ne-am concentrat atenția doar asupra aflării de informații legate de clase, și nu asupra aflării de informații legate de obiectele care reprezintă instanțe ale unor clase. În concluzie, anumite metode nu dau același rezultat cu cel obținut în urma aplicării lor pentru obiecte.

Structura acestei clase este următoarea:

```
class _Class{
  var $classname;

  function _Class($clazz = NULL){...}
  function getType(){...}
  function &getParentClass(){...}
  function &getAncestors(){...}
  function &getAncestorsNames(){...}
  function &getOwnMethodsNames(){...}
  function &getAllMethodsNames(){...}
  function &getOwnDeclaredFields(){...}
  function &getAllDeclaredFields(){...}
}
```

În cadrul acestei implementări, datorită faptului că în cadrul limbajului *PHP* nu există o clasă din care să fie derivate toate celelalte, cum este cazul clasei `Object` din limbajul *Java*, am considerat că părintele tuturor claselor este o clasă virtuală al cărei nume este `"(null)"`, iar, ca o excepție, părintele clasei virtuale `(null)` este tot clasa `(null)`.

În continuare vom detalia clasa `_Class` și vom da câteva exemple de utilizare a acesteia.

Proprietatea classname

Această proprietate reține numele clasei pentru care se efectuează toate operațiile.

Constructorul _Class

Constructorul `_Class` are un singur parametru care este opțional. Acesta poate fi de tip șir de caractere sau de tip obiect.

În cazul în care parametrul este de tip șir de caractere se va verifica existența clasei cu numele dat de acest parametru și dacă o astfel de clasă nu există și numele ei este diferit de `"(null)"` este generată o eroare.

În cazul în care parametrul este de tip obiect, se va utiliza numele clasei care reprezintă numele acestui obiect.

Dacă parametrul lipsește sau are una dintre valorile `NULL` sau `"(null)"` atunci se va considera pentru prelucrările ulterioare clasa virtuală cu numele `"(null)"`.

În continuare vă prezentăm codul sursă al acestui constructor:

```
function _Class($clazz = NULL){
  if (is_string($clazz)){
    if (strtolower($clazz) == "(null)")
      $this->classname = NULL;
    else{
      $this->classname = $clazz;
      if (!class_exists($this->classname))
        trigger_error("Class $clazz not found!",
                                    E_USER_ERROR);
```

```
    }
  }
  elseif (is_object($clazz))
    $this->classname = get_class($clazz);
  elseif (is_null($clazz))
    $this->classname = NULL;
  else
    trigger_error("Invalid parameter type",
                                  E_USER_ERROR);
}
```

Metoda _Class->getType()

Această metodă returnează numele clasei asupra căreia se efectuează operațiile date de celelalte metode, nume care este dat de variabila `classname`.

Codul sursă al acestei metode este următorul:

```
function getType(){
  if (is_null($this->classname))
    return "(null)";
  return $this->classname;
}
```

Metoda _Class->getParentClass()

Această metodă returnează o referință la un obiect de tipul `_Class` al cărei constructor a fost apelat pentru părintele clasei curente.

Codul sursă al acestei metode este următorul:

```
function &getParentClass(){
  if (is_null($this->classname))
    return new _Class(NULL);
  $parent = get_parent_class($this->classname);
  if ($parent)
    return new _Class($parent);
  else
    return new _Class(NULL);
}
```

Metoda _Class->getAncestors()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tipul `_Class` și reprezintă strămoșii clasei curente.

Această metodă a fost implementată folosind un ciclu repetitiv **while**. Vom folosi variabila `$cls` care la început este inițializată cu numele clasei curente și, parcurgând în sus ierarhia de clase, pornind de la clasa curentă și până la întâlnirea valorii **NULL** vom completa elementele tabloului a cărei referință va fi returnată ca rezultat.

Codul sursă al acestei metode este:

```
function &getAncestors(){
  if (is_null($this->classname))
    return array(new _Class(NULL));
  $ancestors = array();
  $cls = $this->classname;
  while($cls = get_parent_class($cls))
    $ancestors[] = new _Class($cls);
  $ancestors[] = new _Class(NULL);
  return $ancestors;
}
```

Metoda _Class->getAncestorsNames()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tip șir de caractere și reprezintă numele strămoșilor clasei curente.

Diferența dintre această metodă și cea precedentă constă în faptul că în tabloul creat se vor stoca doar numele claselor fără a se mai apela constructorul clasei `_Class`.

Codul sursă al acestei metode este:

```
function &getAncestorsNames(){
  if (is_null($this->classname))
    return array("(null)");
  $ancestors = array();
  $cls = $this->classname;
  while($cls = get_parent_class($cls))
    $ancestors[] = $cls;
  $ancestors[] = "(null)";
  return $ancestors;
}
```

Metoda _Class->getAllMethodsNames()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tip șir de caractere și reprezintă numele metodelor pe care clasa curentă le conține.

Tabloul construit de metoda `getAllMethodsNames` conține și numele metodelor clasei curente care au fost moștenite de la clasa părinte.

Codul sursă al acestei metode este:

```
function &getAllMethodsNames(){
  if (is_null($this->classname))
    return array();
  return get_class_methods($this->classname);
}
```

Metoda _Class->getOwnMethodsNames()

Această metodă returnează o referință către un tablou unidimensional ale cărui elemente sunt de tip șir de caractere și reprezintă numele metodelor care au fost definite în cadrul clasei curente.

Construirea acestui tablou se realizează astfel: se determină toate metodele pe care le posedă clasa curentă sub forma unui tablou unidimensional primit ca rezultat în urma apelului funcției `get_class_methods`, se determină toate metodele pe care le posedă părintele clasei curente sub forma unui tablou unidimensional și se returnează o referință spre un tablou unidimensional care conține diferența dintre cele două tablouri determinate.

Codul sursă al acestei metode este:

```
function &getOwnMethodsNames(){
  if (is_null($this->classname))
    return array();
  $clsm = get_class_methods($this->classname);
  $pcls = get_parent_class($this->classname);
  if (!$pcls) return $cls;
  $pclsm = get_class_methods($pcls);
  return array_diff($clsm, $pclsm);
}
```

Metoda _Class->getAllDeclaredFields()

Această metodă returnează o referință spre un tablou unidimensional care conține valorile implicite ale câmpurilor implicite definite în cadrul clasei curente. Acest tablou este obținut în urma apelului funcției `get_class_vars` pentru a determina numele proprietăților implicite definite în cadrul clasei curente. Acest tablou conține și proprietățile moștenite de la clasa părinte.

Codul sursă al acestei metode este următorul:

```
function &getAllDeclaredFields(){
  if (is_null($this->classname))
    return array();
  return get_class_vars($this->classname);
}
```

Metoda _Class->getOwnDeclaredFields()

Această metodă returnează o referință spre un tablou unidimensional care conține valorile implicite ale câmpurilor implicite definite în cadrul clasei curente fără a lua în considerare proprietățile moștenite de la clasa părinte.

Tabloul a cărui referință se returnează se construiește folosind diferența dintre tabloul care conține proprietățile clasei curente și tabloul care conține proprietățile clasei părinte, ca în cazul metodei `getOwnMethodsNames`.

Codul sursă al acestei metode este următorul:

```
function &getOwnDeclaredFields(){
  if (is_null($this->classname))
    return array();
  $vars = get_class_vars($this->classname);
  $pcls = get_parent_class($this->classname);
    if (!$pcls) return $vars;
  $pvars = get_class_vars($pcls);
  return array_diff($vars, $pvars);
}
```

Exemplu

În continuare vă prezentăm un exemplu de utilizare a clasei `_Class` care este definită în fișierul *class_introspector.php*.

Script-ul următor afișează pentru fiecare clasă, încărcată în mediul de execuție al interpretorului *PHP*, numele strămoșilor, metodele proprii și valorile implicite ale proprietăților acestora. În figura 2 poate fi observat efectul execuției acestui script.

```
<?PHP
  include "class_introspector.php";

  function printAncestors($b){
    $a=$b->getAncestors();
    for($i = 0;$i < sizeof($a);$i++)
      echo $a[$i]->getType().' ';
  }

  function printMethods($b){
    $a = $b->getOwnMethodsNames();
    for($i = 0;$i < sizeof($a);$i++)
      echo $a[$i].' ';
  }

  function printFields($b){
    $a = $b->getOwnDeclaredFields();
    foreach($a as $key => $keyv)
      echo "'".$key." = (".gettype($keyv).")"
                                  .$keyv."' ";
  }
?>
<HTML>
  <HEAD>
    <TITLE>Exemplu de utilizare _Class</TITLE>
  </HEAD>
  <BODY><PRE>
  <?PHP
  $a = get_declared_classes();
  for($i = 0;$i < sizeof($a);$i++){
    $b = new _Class($a[$i]);
    echo "<B>".$b->getType()."</B><BR>";
    echo "  <B>Strămoși: </B> ";
    printAncestors($b);
    echo "<BR>";
    echo "  <B>Metode proprii:</B> ";
    printMethods($b);
    echo "<BR>";
    echo "  <B>Proprietăți proprii
                          predefinite:</B> ";
    printFields($b);
    echo "<BR>";
    echo "<BR>";
  }
  ?>
  </PRE></BODY>
</HTML>
```

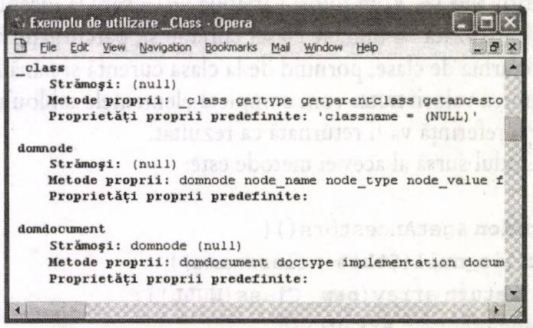


Figura 2: Exemplu de utilizare a clasei _Class

Cel mai tare ghid IT
cele mai-noi știri
-noi produse
-tari teste
-bune soluții-hard
-soft
-internet
Din aprilie,
citește noul PC Magazine
DELL
Vrei să
PC MAGAZINE
ROMÂNIA
www.pcmagazine.ro
SOLUȚII
linux
Tempus
office
Automatizarea completărilor în Excel
grafică
Crearea unui text animat
comunicații
Despre MIDlet-uri și integrarea cu alte sisteme
INTERNET PRO
Securitatea Rețelei [5]: atacurile DoS
JSP: Java și mai aproape de noi
Transmisia prin fibra optică
GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR
LG
Life's Good
Monitor LG
Flatron LCD L1720P
calitatea imaginii, accesibilitate, mobilitate, design atractiv
FLATRON
Windows
ÎNCOTRO
Contract JACK

Publicarea acestei reviste este posibilă datorită companiilor de mai jos care sprijină învățământul românesc de informatică

SPONSORI DE PLATINĂ

SPONSOR DE AUR

Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-266.516

Gfo
Mai 2004
Vol. 14/5
Lei 30.000

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

Probleme

Plus

focus
OPTIMIZARE folosind operații pe biți

tehno
KNOPPIX Linux-versiune fără instalare

concurs
TOP 50 – după 50 de runde • Bilanț

...bună introducere din lume în rețelele de calculatoare
...t actualizată pentru tehnologiile cheie de mâine
EDIȚIA A PATRA
Rețele de calculatoare
ANDREW S. TANENBAUM
Cea mai bună introducere în arhitectura calculatoarelor
- complet actualizată, reflectă ultimele schimbări din dinamica
lume a calculatoarelor
Andrew S. Tanenbaum
ORGANIZAREA STRUCTURATĂ A CALCULATOARELOR
Ediția a IV-a
SISTEME DE OPERARE MODERNE
EDIȚIA A DOUA
Andrew S. Tanenbaum
Această carte de referință, bazată pe exemple concrete,
oferă informații practice de primă mână pentru înțelegerea
și construirea sistemelor de operare moderne.
THOMAS H. CORMEN CHARLES E. LEISERSON RONALD R. RIVEST
INTRODUCERE ÎN
ALGORITMI
Cartea tratează aspecte teoretice și practice, legate de proiectarea
algoritmilor, fiind binevenită elevilor, studenților și profesorilor
CĂRȚILE ESENȚIALE
...cărțile care îți însoțesc cariera
Comandă-le pe toate și le iei cu 20% mai ieftin.
Iar dacă ești student, cu 30%.
BYBLOS
Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85
e-mail: dpastrama@byblos.ro, www.byblos.ro
Librăria Eminescu, Universitate, la subsol
tel: 021-312.72.65, 0721-927.499

Vol. 14, Nr. 5 /2004

gazeta de informatică

Redacția:
Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Director Bursele Agora:
Mihai Scorțaru
Redactor-șef:
Claudiu Soroiu
Redactor-șef adjunct:
Carmen Șușman
Redactori:
prof. *Eugen Ionescu*
Mihai Stroe
Colaboratori:
Cosmin-Silvestru Negrușeri
Siviu Niculiță

Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
400091, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/592.422, 0265/266.516
Fax: 0265/266.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359
Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Vânzare reclamă:
Mihai Bucuroiu
Denis Marioara
Marius Comărniceanu
Difuzare:
Florin Dinu
Florin Oprescu
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Sumar

probleme.. PAG. 5

Olimpiada NAȚIONALĂ de Informatică 2004

~ enunțurile problemelor ~
~ propuse spre rezolvare ~
~ elevilor de gimnaziu ~
~ și pentru lărgirea ~
~ lotului național ~

Concursul de programare ȘTEFAN ODOBLEJA

~ enunțurile problemelor ~
~ propuse spre rezolvare ~

mate.. PAG. 19

Teoria jocurilor: numerele SPRAGUE GRUNDY

Cosmin-Silvestru NEGRUȘERI

~ elemente teoretice ~
~ legate de rezolvarea ~
~ jocurilor pe calculator ~

concurs.. PAG. 24

Despre rundele #41-#50

Top 50 după 50 de runde

Bilanț

focus.. PAG. 32

Optimizarea programelor folosind OPERAȚII PE BIȚI

Cosmin-Silvestru NEGRUȘERI

~ probleme rezolvate ~
~ folosind optimizări ~
~ la nivelul operațiilor ~
~ de bază efectuate~
~ pe biți ~

techno.. PAG. 37

KNOPPIX Linux

Silviu NICULIȚĂ

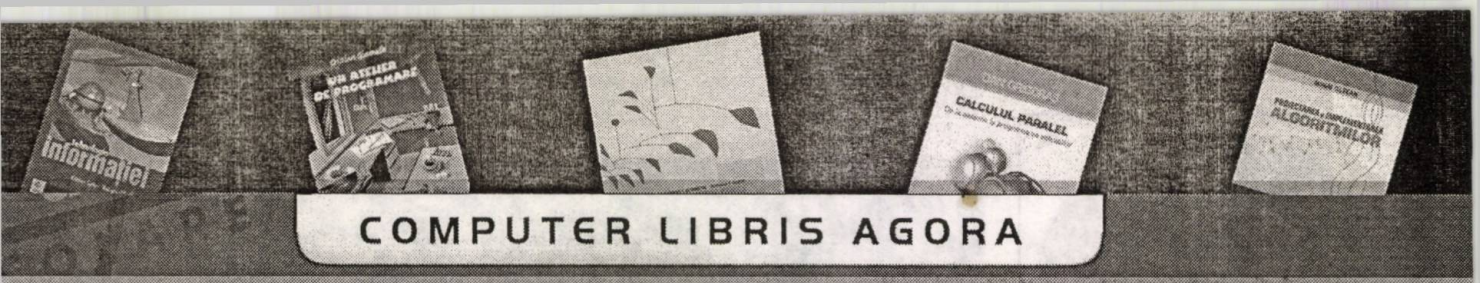
~ distribuție de linux ~
~ care nu necesită ~
~ instalare ~

internet.. PAG. 41

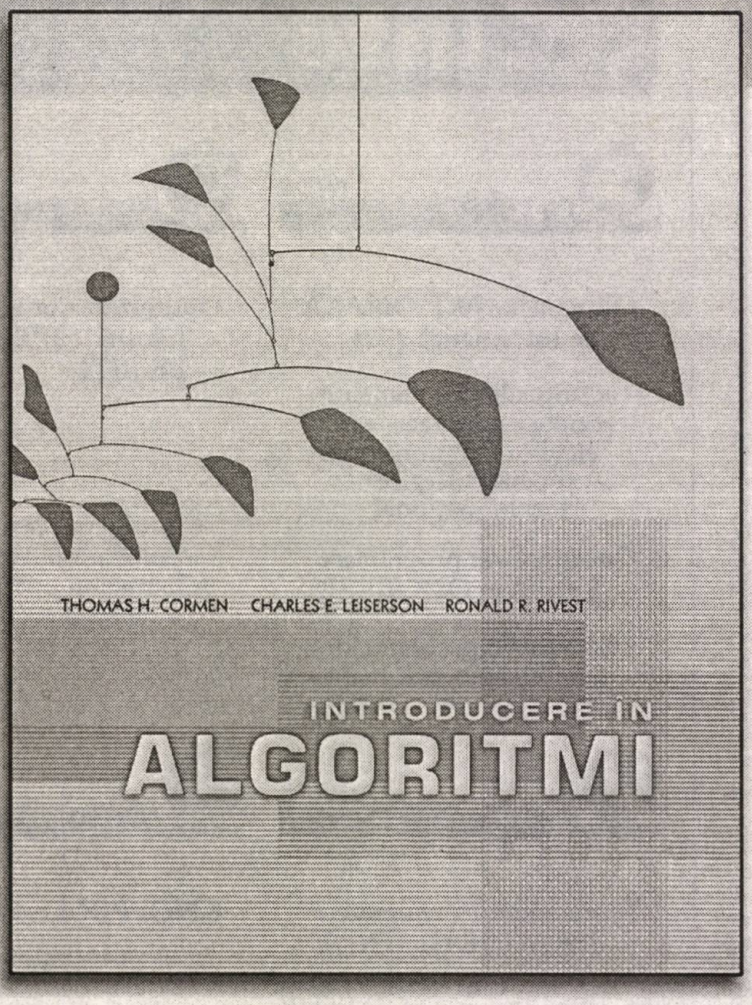
Pagini WEB cu PHP 5

Mihai SCORȚARU
Claudiu SOROIU

~ facilități introduse de ~
~ interpretorul PHP 5 ~



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paralleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Device error: (A)bort, (R)etry, (F)ail...

Cu pași mărunți spre ROBOTIZARE

Sunt tare curios să aflu ce considerați că vor conține rândurile înșirate de-a lungul acestei pagini citind doar titlul...

Ei bine, îmi pare rău să vă spun că s-ar putea să vă fi înșelat în privința conținutului acestui editorial.

Totuși, să vedem ce semnifică cuvântul *robot*. ***Un robot este o mașină realizată de om cu scopul de a îndeplini anumite sarcini complexe și de cele mai multe ori în mod repetat.*** Există mai multe tipuri de roboți pe care nu are rost să le enumăr acum, deoarece nu ar avea loc pe această pagină, sau dacă ar avea loc, nu le-ar putea citi nici o persoană fără ajutorul unui *robot*.

Cât timp suntem tineri, majoritatea persoanelor mai în vârstă ne încurajează spunându-ne că suntem la început de drum și că încă nu cunoaștem greutățile vieții. Sunt perfect de acord cu faptul că suntem la început de drum. Tot timpul suntem la începutul unui drum pe care însă nu-l cunoaștem, dar din greutățile vieții fiecare a cunoscut câte ceva, poate că unii mult mai multe decât cei care ne încurajează.

Fiecare dintre noi, pentru a se integra în societate și pentru a trăi decent trebuie să-și caute un loc de muncă și să muncească. Este problema fiecăruia cum își va găsi locul de muncă și ce compromisuri va face pentru a reuși să se descurce. Această acțiune constituie una dintre greutățile vieții. De cele mai multe ori nu ne găsim locul de muncă pe care l-am dorit din totdeauna și poate nici nu va fi în domeniul în care ne-am perfecționat.

Setea de cunoaștere, dorința de a avea un viitor mai bun, ... ne fac să ajungem, uneori să executăm sarcini ca niște roboți. Pe perioade scurte de timp devenim persoane insensibile care *tac și fac*.

Unii dintre noi fac anumite lucruri din, pasiune alții pur și simplu fac considerând că este necesar, dacă nu pentru ei, pentru membrii familiilor lor.

Această luptă pentru mai bine durează uneori destul de mult, pentru unele persoane chiar o viață.

Dacă suntem cu *capul pe umeri* găsirea unui loc de muncă nu este o problemă prea mare. Problema apare în momentul în care activitățile pe care le efectuăm ne epuizează. Sau, dacă munca pe care o depunem nu este ceea ce ne-am dorit-o și se ivește o activitate care ne place, ne-o mai punem pe cap și pe aceasta și ajungem în situația în care nu mai avem timp de noi. Dimineața mergem la serviciu, ne întoarcem seara obosiți și ne odihnim până a doua zi când începem de la capăt. Continuând așa, ajungem uneori insensibili cu persoanele apropiate nouă și începem să ne pierdem ceea ce ne este cel mai drag nouă și ceea ce ne face să simțim că trăim, deși în adâncul sufletului nostru dorim ca totul să iesă bine.

Pentru unii dintre noi, aceste acțiuni sunt catalogate drept compromisuri, dar revin asupra definiției cuvântului *robot*.

Ceea ce ne separă pe noi de mașini sunt sentimentele. *Un robot este și o persoană lipsită de sentimente (sentimente pe care le-a avut și le-a suprimat de-a lungul anilor) care execută eficient diferite sarcini.*

Mulți dintre noi se laudă că ei știu ce este bine și ce este rău. S-ar putea să aibă dreptate. Problema însă este mult mai complicată. Știm foarte bine ce este "bine", aspirăm la el, dar nu reușim (de cele mai multe ori) să îl și atingem. Haideți să identificăm binele cu albul și răul cu negrul. Nimic din ceea ce facem nu este nici alb nici negru. Efectul tuturor acțiunilor, gândurilor și trăirilor noastre sunt nuanțe de gri mai deschise sau mai închise. Nu există nimic alb sau negru. Dacă ar exista, acea culoare ar constitui soluția omenirii pentru a atinge perfecțiunea.

Societatea ne oferă posibilitatea de a trece mai ușor prin viață, dar noi, vrem, uneori, mult mai mult decât ni se oferă.

În condițiile în care ne străduim să fie bine, dacă nouă ni se pare că ceva este un gri deschis, atunci trebuie să ne gândim că undeva s-ar putea să fi făcut o greșeală pe care nu am observat-o și alții vor vedea același lucru mai închis. Albul nu poate exista fără negru. Orice lucru sau acțiune are părți bune și părți mai puțin bune.

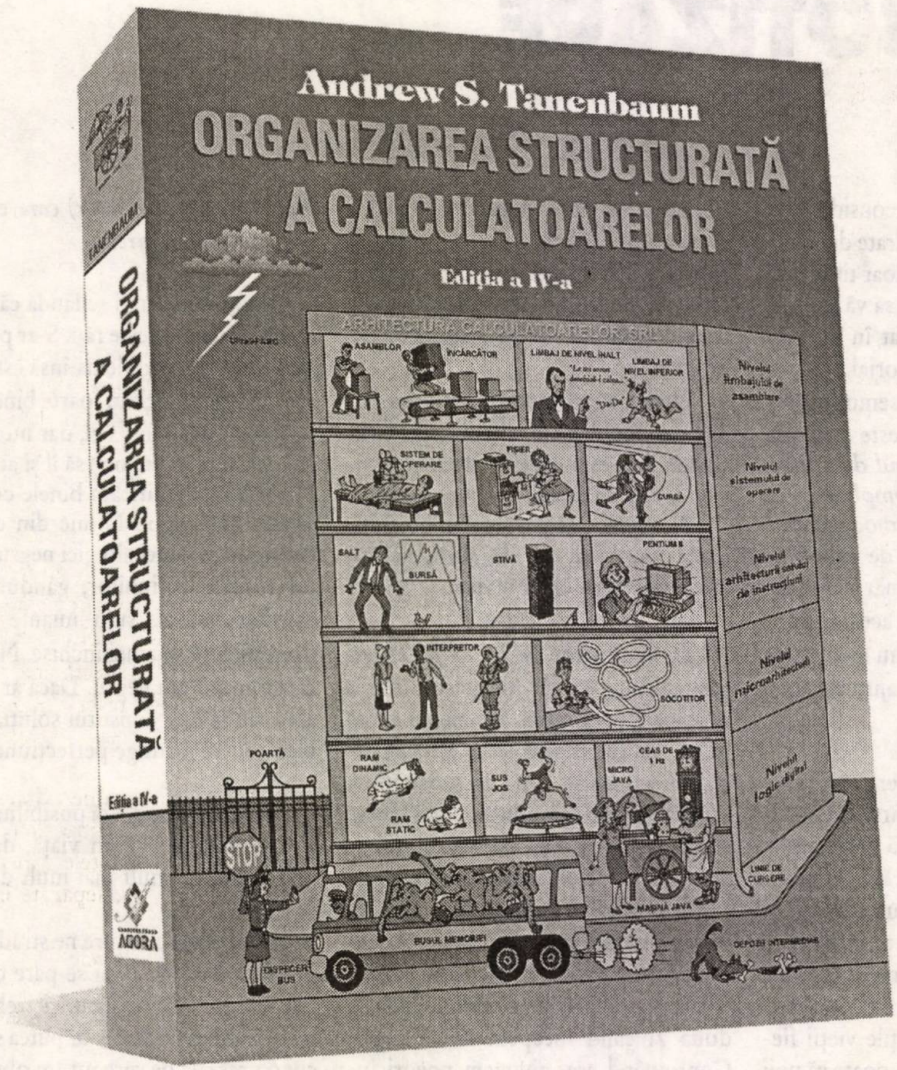
Indiferent ce decizii luăm și indiferent de situațiile cărora trebuie să le facem față, trebuie să ne gândim și nu trebuie să cedăm când trebuie să facem ceva ce considerăm că ne depășește.

Viața este un joc pe care nu poți considera că l-ai pierdut dacă nu ai încercat totul, așa că (R)etry...

Claudiu Soroiu

Cine are carte, are parte!

...de înțelegerea calculatoarelor!



Comandă-o acum și dacă ești student

primești și o REDUCERE semnificativă!

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65, 0721-927.499

Buzău, 2-9 aprilie 2004

Olimpiada NAȚIONALĂ de Informatică 2004

Vă prezentăm în continuare enunțurile celor 18 de probleme propuse spre rezolvare la ediția din acest an a Olimpiadei Naționale de Informatică pentru elevii de gimnaziu și cele pentru lărgirea lotului național.

Clasa a V-a

P040501: Produs

Ionică, elev în clasa a V-a la o școală generală din *Buzău* este pasionat de matematică și de telefoane mobile. Într-un moment de plictiseală când se uită în agenda telefonică la un număr de telefon format din maxim 10 cifre (prima cifră fiind diferită de 0) se gândește la următoarea problemă: *Între care cifre ale numărului de telefon este bine să pună operatorul de înmulțire pentru a obține două numere cu produsul maxim?*

Se cere să se afișeze cele două numere și produsul lor.

Date de intrare

De la *intrarea standard* (tastatură) se va citi numărul de telefon.

Date de ieșire

La *ieșirea standard* (ecran) se vor furniza cele două numere și valoarea produsului unul sub altul. Cele două numere se vor afișa în ordinea cerută.

Restricții și precizări

- numărul de telefon este un număr natural cu cel puțin două cifre și mai mic sau egal cu 2100000000.

Exemplu

Intrare standard	*Ieșirea standard*
2301	2 301 602

Timp de execuție: 1 secundă/test

P040502: Doar o notă să-ți mai pun...

Astăzi, doamna profesoară va trece în catalog notele obținute la lucrarea de control de către elevii săi din clasa a V-a. Înainte de acest lucru dorește să afle care este nota (notele) care apare de cele mai multe ori (are cea mai mare frecvență) și notele care nu s-au acordat la această lucrare.

Se cere să se determine și să se afișeze notele care apar de cele mai multe ori, precum și notele care nu s-au obținut la lucrare de către nici un elev.

Date de intrare

De la *intrarea standard* (tastatură) se va citi un număr natural n, care reprezintă numărul de elevi din clasă și cele n note obținute de elevi la lucrarea de control.

Datele vor fi introduse în ordinea specificată mai sus, iar după fiecare valoare numerică se va tasta <Enter>.

Date de ieșire

La *ieșirea standard* (ecran) se vor furniza pe o linie notele care apar de cele mai multe ori, separate între ele prin spațiu. Pe următoarea linie se vor furniza notele care nu s-au obținut la lucrare de către nici un elev, separate între ele prin spațiu.

Nu se vor afișa alte valori în afară de cele cerute.

Restricții și precizări

- $1 \le n \le 25$;
- toate notele sunt numere naturale cuprinse între 1 și 10.

Exemplu

Intrare standard	*Ieșirea standard*
4 4 5 7 5	5 1 2 3 6 8 9 10

Timp de execuție: 1 secundă/test

P040503: Prințesa

În regatul numerelor naturale, prințesei *Prima* i-a venit sorocul să se mărite. Solii regelui au dat de veste că orice pre-

tendent la mâna prințesei trebuie să fie număr prim. Sfetnicii regelui au găsit însă H rude ale prințesei care nu pot să-i devină soț.

Cunoscându-se care sunt aceste rude, se cere să se afle cei dintâi K pretendenți posibili, în ordinea descrescătoare a lor.

Date de intrare

De la *intrarea standard* (tastatură) se vor citi numerele K și H și rudele prințesei. Datele se vor citi în ordinea precizată anterior, fiecare valoare numerică fiind urmată de <Enter>.

Date de ieșire

La *ieșirea standard* (ecran) se vor furniza cei K pretendenți, pe același rând, separați între ei printr-un spațiu.

Restricții

- $1 \le H \le 10$;
- $1 \le K \le 30$;
- rudele prințesei sunt numere naturale mai mici sau egale cu 100.

Exemplu

Intrare standard	*Ieșirea standard*
K=4 H=5 4 3 12 29 7	13 11 5 2

Timp de execuție: 1 secundă/test

Clasa a VI-a

P040504: Spider

Spider este un păianjen care trăiește în casa unui programator. De la acesta *Spider* a preluat pasiunea pentru numere și pentru programe. Astfel stând lucrurile, *Spider* a hotărât să nu-și mai țească pânza în mod tradițional, ci să folosească informațiile aflate de la programator, abordând și un stil de lucru metodic. Prin urmare, *Spider* procedează astfel:

- alege n puncte așezate în cerc și le numerotează de la 1 la n (în sensul acelor de ceasornic);
- calculează distanțele dintre oricare două puncte obținând doar numere naturale distincte;
- alege un punct de plecare k;
- stabilește următoarea regulă pe care o va respecta când va țese pânza: în fiecare zi va țese câte un fir: dacă numărul zilei este impar, atunci țese firul de la punctul în care se află la punctul următor (de asemenea în sensul acelor de ceasornic, iar după punctul numerotat cu n urmează punctul numerotat cu 1), iar dacă numărul zilei este par *Spider* țese un fir între punctul în care se află și punctul în care ajunge sărind un punct;
- se oprește atunci când ar trebui să țească un fir între două puncte între care există deja un fir țesut.

Se cere să se determine numărul de zile necesar pentru a-și țese pânza și punctul în care s-a oprit și lungimile firelor țesute împreună cu capetele lor, în ordinea descrescătoare a lungimilor lor. Capetele firelor vor fi afișate în ordine crescătoare.

Date de intrare

Datele se citesc de la *intrarea standard* (tastatură).

Pe prima linie se află un număr natural n, reprezentând numărul de puncte alese.

Pe cea de-a doua linie se află un număr natural k, reprezentând punctul de plecare.

Pe fiecare dintre următoarele n linii se află n numere naturale separate între ele prin spații, astfel că, cel de-al j-lea element de pe cea de-a i-a linie (d_{ij}) reprezintă distanța dintre punctele numerotate cu i, respectiv j găsită de *Spider*.

Date de ieșire

La *ieșirea standard* (ecran) se vor scrie pe prima linie două numere x și p, separate printr-un spațiu, care reprezintă numărul de zile, respectiv punctul în care s-a oprit *Spider*.

Pe următoarele p linii se vor scrie trei numere l, $c1$, $c2$ separate între ele prin spațiu, care reprezintă lungimea unui fir și capetele lui. Capetele firelor se vor scrie în ordine crescătoare, iar datele pe aceste p linii se vor scrie în ordine descrescătoare a lungimilor.

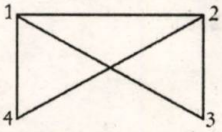
Restricții și precizări

- $1 \le n \le 100$;
- $1 \le k \le n$;
- $0 \le d_{ij} \le 50.000$, $\forall 1 \le i, j \le n$.

Exemplu

Intrare standard	*Ieșirea standard*
4 2 0 5 8 7 5 0 3 10 8 3 0 4 7 10 4 0	5 1 10 2 4 8 1 3 7 1 4 5 1 2 3 2 3

Explicație



În prima zi *Spider* pleacă din punctul 2 și țese un fir până la punctul 3. În ziua a doua, din punctul 3 țese un fir până la punctul 1 (sare punctul 4). În ziua a treia din punctul 1 țese un fir până la punctul 2. În ziua a patra din punctul 2 țese un fir până la punctul 4 (sare punctul 3). În ultima zi din punctul 4 țese un fir până la punctul 1 și se oprește.

Timp de execuție: 1 secundă/test

P040505: Numere prime "apropiate"

Pentru temă, *Alina* trebuia să-și verifice cunoștințele legate de numere prime.

Alina va citi (chiar în această ordine) o valoare N care semnifică numărul numerelor naturale care vor fi prelucrate și o valoare C care poate fi 1 sau 2. În continuare va citi cele N numere naturale. Fiecare număr natural A citit, va fi prelucrat astfel:

- dacă A este număr prim, el va fi afișat nemodificat;
- dacă A nu este prim, se va determina cel mai apropiat număr prim care va fi afișat. Considerăm cel mai apropiat număr de valoarea A, cel pentru care modulul diferenței dintre număr și valoarea A este cel mai mic. Fie X cel mai mare număr prim mai mic decât A, și Y cel mai mic număr prim mai mare decât A. Dacă cele două numere sunt la fel de "apropiate" de A atunci, dacă $C = 1$ se va afișa X și dacă $C = 2$ se va afișa Y.

Date de intrare

Datele se citesc de la *intrarea standard* (tastatură). Pe prima linie se află două numere N și C, separate între ele printr-un singur spațiu, reprezentând numărul de numere care trebuie prelucrate, respectiv valoarea C care poate fi 1 sau 2, valoare care va decide dacă se afișează valoarea primă cea mai apropiată mai mică, respectiv mai mare decât valoarea corespunzătoare.

Pe cea de-a doua linie se află cele N numere (a_1, a_2, ..., a_n) care trebuie prelucrate, separate între ele prin spațiu.

Date de ieșire

La *ieșirea standard* (ecran) se vor afișa N numere naturale, separate printr-un spațiu, reprezentând numerele prime "apropiate" de cele inițiale.

Restricții și precizări

- $1 \le N \le 150$;
- $3 \le a_i \le 32.700$;
- nu se vor afișa pe ecran alte valori, în afara celor cerute ca rezultate ale problemei.

Exemple

Intrare standard
```
15 1
3 6 8 2 3 5 7 9 2 5 3 7 11 22 21
```

Ieșirea standard
```
3 5 7 2 3 5 7 7 2 5 3 7 11 23 19
```

Intrare standard
```
13 2
3 6 7 11 21 9 5 7 14 29 24 25 16
```

Ieșirea standard
```
3 7 7 11 23 11 5 7 13 29 23 23 17
```

Timp de execuție: 1 secundă/test

P040506: Palindrom

Ionel are de rezolvat la matematică o problemă care presupune calcularea unei expresii care conține paranteze, operații de adunare, scădere, înmulțire, împărțire cu numere naturale.

Problema este că *Ionel* nu are chef să calculeze, astfel încât a început să se joace cu toate numerele din expresie: le-a despărțit în cifrele componente și, fiindcă a auzit de numere palindroame de la un coleg care știa ceva informatică, a încercat să scrie un număr cât mai mare cu cifrele pe care le are la dispoziție și care să fie palindrom.

Deoarece jocul i s-a părut prea simplu a decis ca în cazul în care va folosi o cifră el va trebui să o utilizeze de câte ori apare în expresia aritmetică.

Cunoscând cifrele din expresie, să se determine cel mai mare număr palindrom ce se poate forma respectând regula de mai sus.

Un număr este palindrom dacă citit de la stânga la dreapta are aceeași valoare ca și la citirea de la dreapta la stânga (exemplu: 17271).

Date de intrare

Datele se citesc de la *intrarea standard* (tastatură). Pe prima linie se află un număr natural n care reprezintă numărul de cifre. Pe cea de-a doua linie se află cele n cifre separate între ele printr-un spațiu.

Date de ieșire

La *ieșirea standard* (ecran) se va afișa, pe o singură linie, numărul cerut.

Restricții și precizări

- $1 \le n \le 10.000$;
- nu se vor afișa alte valori numerice pe ecran, în afara numărului cerut și nu se vor afișa spații între cifrele numărului.

Exemple

Intrare standard	*Ieșirea standard*
4 2 4 5 2	252

Intrare standard	*Ieșirea standard*
7 2 2 3 5 3 3 2	333

Timp de execuție: 1 secundă/test

Clasa a VII-a

P040507: V-uri

Se consideră un tablou bidimensional care are m linii și n coloane. Se numește traseu în V o parcurgere prin elementele tabloului astfel:

- se pleacă întotdeauna de la un element de pe linia 1 a tabloului, se ajunge în final la un alt element de pe linia 1 a

tabloului, trecând prin cel puțin trei elemente, fără a trece printr-un element de mai multe ori;
- parcurgerea elementelor tabloului se face în forma unei singure litere *V* ca în figura din exemplu, de la element putându-se trece doar la un alt element imediat vecin pe diagonală.

Fiecare element al tabloului conține valori întregi. La parcugerea traseului se calculează suma elementelor de pe traseu.

Determinați traseul în *V* care conține cea mai mare sumă. În cazul în care există mai multe trasee cu aceeași sumă, se va alege traseul care parcurge cele mai puține celule. Dacă și în acest caz există mai multe soluții, se alege traseul cel mai din stânga (cel cu indicele coloanei de pornire cel mai mic).

Date de intrare

Fișierul de intrare **v.in** conține pe prima linie două numere naturale *m* și *n*, separate între ele printr-un singur spațiu, reprezentând numărul de linii și numărul de coloane ale tabloului.

Pe următoarele *m* linii se află câte *n* numere, separate între ele prin spațiu, care reprezintă valorile elementelor unei linii din tablou.

Date de ieșire

Fișierul de ieșire **v.out** va conține o singură linie pe care se vor afla trei numere naturale, separate între ele prin spațiu, reprezentând suma valorilor elementelor din tablou pentru traseul ales, coloana de pornire și linia pe care se află vârful *V*-ului.

Datele din fișierul de ieșire trebuie să fie în ordinea specificată, adică sumă, coloană, linie.

Restricții și precizări

- $1 \le m, n \le 100$;
- *valorile elementelor tabloului* sunt numere întregi cuprinse între -60.000 și 60.000;
- conform datelor de intrare, suma valorilor elementelor din tablou pentru orice traseu ales nu depășește 1.000.000.000.

Exemplu

v.in

```
5 9
3 4 12 4 6 7 9 5 12
0 4 5 7 9 -5 1 1 5
0 98 34 0 1 7 7 1 1
6 7 8 -9 0 2 3 5 22
47 62 31 55 0 83 23 77 10
```

v.out

```
54 1 3
```

Conform datelor de mai sus se pot crea și alte trasee, dar care au suma mai mică decât cea din rezultat:

3	4	12	4	6	7	9	5	12
0	4	5	7	9	-5	1	1	5
0	98	34	0	1	7	7	1	1
6	7	8	-9	0	2	3	5	22
47	62	31	55	0	83	23	77	10

Timpi de execuție: 1 secundă/test

P040508: Jetoane

Ionel este elev în clasa a IV-a și părinții s-au gândit la o metodă eficientă prin care el să învețe numerele mari și modul lor de formare. În acest scop i-au cumpărat un joc cu jetoane. Pentru fiecare cifră există câte 10 jetoane inscripționate pe una din fețe cu cifra respectivă. Ionel poate forma numere prin așezarea jetoanelor unul lângă altul. Părinții îi cer lui *Ionel* să formeze numere, pe rând, care să aibă suma cifrelor *S* și să fie mai mici decât 10^a. Deoarece este dificil să-l urmărească în formarea numerelor, părinții doresc să știe câte astfel de numere distincte știe *Ionel* să formeze.

Câte numere distincte a format *Ionel*?

Date de intrare

Fișierul de intrare **jetoane.in**, conține pe prima linie numărul *S*, iar pe cea de-a doua linie numărul *a*.

Date de ieșire

În fișierul **jetoane.out** se va scrie pe un singur rând numărul cerut.

Restricții și precizări

- $1 \le S < 10$;
- $1 \le a < 10$;
- *S* și *a* sunt numere naturale.

Exemple

jetoane.in

```
2
3
```

jetoane.out

```
6
```

jetoane.in

```
5
5
```

jetoane.out

```
126
```

Explicație

În cazul primului exemplu ($S = 2$ și $a = 3$) *Ionel* formează numerele: 2, 11, 20, 101, 110, 200.

Timpi de execuție: 1 secundă/test

P040509: Neo

Neo, eroul din filmul *Matrix*, pentru a-și învinge adversarii are nevoie de energie. El a descoperit în matrice o zonă care

cuprinde locații consecutive numerotate de la 1 la N, fiecare dintre ele generând o anumită cantitate de energie, pozitivă sau negativă, cunoscută.

Neo are posibilitatea să pătrundă în această zonă printr-o singură locație și poate părăsi zona o singură dată. El, odată intrat în zona energetică, trebuie să străbată locațiile în ordinea crescătoare a numărului lor, iar energia acumulată într-o locație multiplică de atâtea ori energia deja acumulată de *Neo*.

De asemenea, dacă într-o locație se află energie negativă, iar *Neo* a acumulat până în acea locație energie pozitivă, toată energia lui se transformă în energie negativă, dar dacă în drumul său *Neo* întâlnește o altă locație cu energie negativă, iar energia deținută de el este tot negativă, toată energia lui devine pozitivă.

Energia inițială a lui *Neo* are valoarea 1.

Ajutați-l pe *Neo* să descopere locația în care trebuie să intre și locația în care trebuie să părăsească zona energetică pentru a acumula cea mai mare cantitate de energie pozitivă posibilă.

Date de intrare

Fișierul de intrare **NEO.IN** conține pe prima linie numărul locațiilor energetice, iar pe cea de-a doua linie conține un șir de numere întregi, separate între ele printr-un spațiu, reprezentând cantitățile de energie ale locațiilor.

Date de ieșire

Pe prima linie a fișierului de ieșire **NEO.OUT** se va scrie cantitatea de energie acumulată de *Neo*, iar pe cea de-a doua linie se vor scrie două numere separate între ele printr-un singur spațiu, care reprezintă numărul de ordine al locației în care *Neo* pătrunde în zona energetică și numărul de ordine al locației în care *Neo* părăsește zona energetică. Dacă nu se poate obține energie pozitivă, cantitatea de energie acumulată și numărul de ordine al locațiilor vor fi 0.

Restricții și precizări

- $0 \le N \le 100$;
- valoarea energiei dintr-o locație este un număr întreg cuprins între -3 și 3;
- produsul tuturor energiilor este un număr întreg cuprins între -2.000.000.000 și 2.000.000.000;
- dacă există mai multe soluții pentru datele de intrare, se va afișa una singură.

Exemple

NEO.IN	NEO.OUT
4 -3 -2 2 -3	12 2 4

NEO.IN	NEO.OUT
1 -2	0 0 0

Timpi de execuție: 1 secundă/test

Clasa a VIII-a

P040510: Zar

Pe tabla de șah de dimensiune $N \times N$, se găsește un zar (care are latura egală cu latura pătratului care definește un pătrat al tablei de șah). Acest zar se poate rostogoli în conformitate cu secvența de comenzi care se citește dintr-un fișier, fiecare caracter citit având semnificația:

- `u` – rostogolire în pătratul de deasupra (*up*);
- `d` – rostogolire în pătratul de dedesubt (*down*);
- `l` – rostogolire în pătratul din stânga (*left*);
- `r` – rostogolire în pătratul din dreapta (*right*).

Se cunosc:

- poziția inițială a zarului dată prin punctul de coordonate (linie, coloană);
- valorile aflate pe fețele zarului date printr-un șir: *sus*, *jos*, *stânga*, *dreapta*, *față*, *spate*, în această ordine;
- secvența de mișcări date este formată din caracterele '*u*', '*d*', '*l*' și '*r*'.

Să se determine sumele de valori pe care le vor avea, în urma secvenței de mișcări, fețele: sus, jos, stânga, dreapta, față, spate (în această ordine), precum și mulțimea valorilor care sunt pe fața de sus a zarului la fiecare mutare a acestuia.

Prima poziție se consideră cea inițială.

Secvența de mișcări a zarului se realizează cât timp zarul printr-o mutare rămâne pe tabla de șah. Dacă în urma unei mutări zarul ar ieși în afara tablei, se vor afișa rezultatele obținute până în acel moment.

Date de intrare

Fișierul de intrare **ZAR.IN** conține pe prima linie un număr natural N care reprezintă dimensiunea tablei.

Cea de-a doua linie conține două numere *lp* și *cp*, separate între ele printr-un singur spațiu, care reprezintă linia inițială, respectiv coloana inițială pentru poziția zarului.

Cea de-a treia linie conține șase numere v_1, v_2, v_3, v_4, v_5 și v_6, separate între ele printr-un spațiu, care reprezintă valorile fețelor zarului și sunt cuprinse între 1 și 6.

Cea de-a patra linie conține un număr natural h care reprezintă numărul de caractere ale secvenței de mutări, iar ultima linie conține caracterele secvenței de mișcări neseparate prin spații.

Date de ieșire

Fișierul de ieșire cu numele **ZAR.OUT** va trebui să conțină pe prima linie șase numere *suma_sus*, *suma_jos*, *suma_stanga* *suma_dreapta*, *suma_fata*, *suma_spate*, separate între ele prin spațiu, cu semnificația suma punctelor de pe fețele specificate, în această ordine.

Pe cea de-a doua linie a fișierului de ieșire se vor afla $h + 1$ valori care reprezintă valorile de pe fața de sus a zarului pentru secvența de mișcări propusă. Pe prima poziție se va afla valoarea inițială.

Restricții și precizări

- $4 \le N \le 100$;
- secvența de mișcări poate avea până la 1.000 de caractere.

Exemple

ZAR.IN

```
8
7 3
1 6 5 2 4 3
19
uuuuuurrrrdddllddduu
```

ZAR.OUT

```
72 68 77 63 67 73
1 4 6 3 1 4 6 5 1 2 6 4 1 3 2 4 6 3 6 4
```

ZAR.IN

```
8
7 3
1 6 5 2 4 3
9
uuuuuuuuu
```

ZAR.OUT

```
25 24 35 14 27 22
1 4 6 3 1 4 6
```

Timpi de execuție: 1 secundă/test

P040511: Sume

Fie n un număr natural nenul. Scrieți un program care să determine n submulțimi disjuncte două câte două de câte n elemente distincte din mulțimea $\{1, 2, ..., n^2\}$, submulțimi pentru care suma elementelor este aceeași.

Date de intrare

Fișierul de intrare **sume.in** conține pe prima linie numărul natural nenul n.

Date de ieșire

Fișierul de ieșire **sume.out** va conține n linii, câte una pentru fiecare submulțime determinată. Pe linia i se află cele n elemente ale submulțimii i, separate prin câte un spațiu.

Restricții și precizări

- $1 \le n \le 100$;
- două submulțimi sunt disjuncte dacă nu au elemente comune;
- soluția nu este unică, puteți afișa orice soluție care respectă condițiile din enunțul problemei;
- ordinea submulțimilor sau a elementelor submulțimii nu contează.

Exemplu

sume.in

```
4
```

sume.out

```
15 10 5 4
3 8 9 14
13 2 7 12
```

Timpi de execuție: 1 secundă/test

P040512: Plăci meteoritice

Pe planeta *UZABU* va cădea o ploaie de meteoriți. Savanții știu că fiecare meteorit este de formă dreptunghiulară, având laturile paralele cu axele de coordonate (și pe planeta *UZABU* axele de coordonate au aceeași semnificație ca și pe Pământ).

Solul planetei este reprezentat prin axa OX. Meteoriții căzuți pe planetă sunt folositori pentru agricultură. Dacă un meteorit în cădere atinge un alt meteorit, atunci amândoi se vor distruge, iar craterul format distruge solul planetei.

Cunoscând care sunt coordonatele plăcilor date prin patru numere (x_1, y_1, x_2, y_2), cu semnificația: (*stânga_sus_x*, *stânga_sus_y*, *dreapta_jos_x*, *dreapta_jos_y*), savanții trebuie să distrugă o parte dintre meteoriți astfel încât ei să nu se suprapună în momentul atingerii solului.

Date de intrare

Fișierul de intrare **METEOR.IN** conține pe prima linie un număr natural N reprezentând numărul de meteoriți.

Pe fiecare dintre următoarele N linii se află câte patru numere separate între ele printr-un spațiu, reprezentând coordonatele *stânga sus* și *dreapta jos* ale unei plăci de formă dreptunghiulară.

Date de ieșire

Fișierul de ieșire **METEOR.OUT** va conține o singură valoare k, reprezentând numărul maxim al plăcilor rămase.

Restricții și precizări

- $0 \le N \le 500$;
- coordonatele fiecărui meteorit sunt numere întregi cuprinse între 0 și 32.000;
- două plăci care prin cădere se lipesc nu se distrug.

Exemplu

METEOR.IN

```
10
5 10 18 2
15 22 27 12
35 30 40 25
43 30 45 25
32 20 45 12
50 20 60 15
20 30 30 25
65 20 82 15
48 13 75 2
78 14 100 3
```

METEOR.OUT

```
6
```

Timpi de execuție: 1 secundă/test

P040513: Invsort

Se consideră un șir de N numere naturale, care trebuie ordonat crescător. Singura operație permisă este să considerați elementele de pe pozițiile $i, i+1, \ldots, j$ (pentru i și j arbitrare, $i<j$), și să inversați ordinea acestor elemente (adică elementul de pe poziția i ajunge pe poziția j, $i+1$ ajunge pe poziția $j-1, \ldots, j$ ajunge pe poziția i). Costul unei astfel de operații este numărul de elemente din subșirul inversat, și anume $j-i+1$.

Scrieți un program care să determine o secvență de operații care ordonează crescător șirul dat și are un cost total cât mai mic (dar nu obligatoriu minim).

Date de intrare

Fișierul de intrare **invsort.in** conține pe prima linie numărul N și apoi N linii cu elementele șirului dat (nu neapărat distincte).

Date de ieșire

Fișierul de ieșire **invsort.out** va conține pe fiecare linie descrierea unei operații. O operație este descrisă prin două numere i și j, separate printr-un spațiu. Aceste numere satisfac $1 \le i < j \le N$.

Restricții și precizări

- $2 \le N \le 32.000$;
- valorile șirului care trebuie ordonat sunt între 0 și 31.999.

Exemplu

invsort.in	invsort.out
5	3 5
1	1 3
0	
1	
1	
0	

Explicație

- prima operație are efectul: 1 0 [1 1 0] → 1 0 0 1 1
- a doua operație are efectul: [1 0 0] 1 1 → 0 0 1 1 1
- costul total este 3 + 3 = 6

Timpi de execuție:

- *Microsoft Windows*: 2 secunde/test
- *Linux*: 0,5 secunde/test

P040514: Peri

Se consideră o matrice dreptunghiulară A cu m linii și n coloane cu valori 0 sau 1, liniile și coloanele fiind numerotate de la 1 la m, respectiv de la 1 la n. Numim dreptunghi de colțuri (x_1, y_1) (x_2, y_2) cu $x_1<x_2$ și $y_1<y_2$ mulțimea elementelor A_{ij} cu $x_1 \le i \le x_2$ și $y_1 \le j \le y_2$. Numim perimetru al dreptunghiului de colțuri (x_1, y_1) (x_2, y_2) mulțimea elementelor A_{ij} pentru care ($i = x_1$ și $y_1 \le j \le y_2$) sau ($i = x_2$ și $y_1 \le j \le y_2$) sau ($j = y_1$ și $x_1 \le i \le x_2$) sau ($j = y_2$ și $x_1 \le i \le x_2$).

Determinați diferența maximă dintre numărul de elemente egale cu 1 și numărul de elemente egale cu 0 aflate pe perimetrul aceluiași dreptunghi, precum și numărul de dreptunghiuri pentru care se obține această diferență.

Date de intrare

Pe prima linie a fișierului de intrare **peri.in** sunt scrise numerele m și n, separate printr-un singur spațiu. Pe următoarele m linii este dată matricea A, numerele de pe aceeași linie fiind separate de câte un spațiu.

Date de ieșire

Fișierul de ieșire **peri.out** va conține o singură linie pe care se află două numere întregi separate printr-un spațiu. Primul număr este diferența maximă dintre numărul de elemente 1 și numărul de elemente 0 de pe perimetrul unui dreptunghi. Al doilea întreg este numărul de dreptunghiuri pentru care diferența dintre numărul de elemente 1 și numărul de elemente 0 de pe perimetru este maximă.

Restricții și precizări

- $1 < m, n \le 250$;
- prin diferență nu se înțelege *diferență în valoare absolută*.

Exemplu

peri.in	peri.out
4 5	4 2
1 0 0 1 0	
0 1 1 0 0	
0 1 0 1 0	
1 1 1 0 1	

Timpi de execuție:

- *Microsoft Windows*: 1 secundă/test
- *Linux*: 0,2 secunde/test

P040515: Trans

În depozitul unei companii de construcții se află N blocuri de piatră, de culoare albă sau neagră. Ele sunt așezate în ordinea 1, 2, ..., N, de la intrarea în depozit către interior.

Blocurile de piatră trebuie să fie transportate pe un șantier de construcții, în ordinea în care ele sunt depozitate, iar pentru aceasta va trebui închiriat un camion de la o companie de transport. Aceasta deține Q tipuri de camioane. Camionul de tipul i ($1 \le i \le Q$) poate transporta maxim K_i blocuri de piatră la un moment dat și pentru un transport se percepe taxa T_i.

Compania de transport este de părere că, pentru a-și păstra clientela, trebuie să impună anumite standarde, indiferent de cât de absurde ar fi, deci impune condiția ca toate blocurile de piatră plasate în camion la un transport să aibă aceeași culoare. Așadar, pentru a fi transportate toate blocurile pe șantier, compania de construcții va alege un camion de un anumit tip, iar camionul va efectua unul sau mai multe transporturi.

Pentru a micșora suma totală plătită, compania de construcții are posibilitatea de a schimba culoarea oricărui bloc de piatră (din alb în negru sau din negru în alb); pentru fiecare bloc i ($1 \le i \le N$) se cunoaște suma S_i care trebuie plătită pentru a-i schimba culoarea.

Pentru fiecare dintre cele Q tipuri de camioane deținute de compania de transport, determinați suma minimă pe care o va plăti compania de construcții pentru a transporta toate cele N blocuri pe șantier.

Date de intrare

Fișierul de intrare **trans.in** conține pe prima linie numărul întreg N, reprezentând numărul de blocuri de piatră din depozit.

Pe fiecare dintre următoarele N linii se află informații referitoare la câte un bloc de piatră. Pe a i-a dintre aceste N linii se găsesc două numere întregi separate printr-un spațiu: C_i și S_i, reprezentând culoarea celui de-al i-lea bloc (C_i este 0 pentru alb și 1 pentru negru) și respectiv suma ce trebuie plătită pentru a-i schimba culoarea (dacă este necesar).

Pe următoarea linie se află numărul natural Q, reprezentând numărul de tipuri de camioane deținute de compania de transport. Pe fiecare dintre următoarele Q linii se află informații referitoare la câte un camion. Pe cea de a i-a dintre aceste Q linii sunt scrise două numere naturale, separate printr-un spațiu K_i T_i, reprezentând numărul maxim de blocuri care pot fi transportate simultan de către un camion de tipul i și respectiv taxa care trebuie plătită pentru fiecare transport efectuat.

Date de ieșire

Fișierul de ieșire **trans.out** va conține Q linii. Pe cea de a i-a dintre aceste linii va fi afișată suma totală minimă plătită de compania de construcții pentru a transporta toate cele N blocuri de piatră, în cazul în care ar închiria un camion de tipul i.

Restricții și precizări

- $1 \le N \le 16.000$;
- $1 \le S_i \le 10.000$;
- $1 \le Q \le 100$;
- $1 \le K_i \le N$;
- $1 \le T_i \le 100.000$.

Exemplu

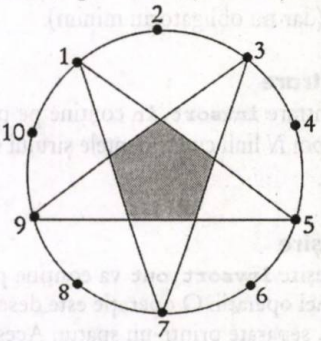
trans.in	trans.out
4	1005
0 2	4
1 3	14
0 10	
1 2	
3	
4 1000	
4 1	
2 5	

Timpi de execuție:

- *Microsoft Windows*: 1 secundă/test
- *Linux*: 0,3 secunde/test

P050326: Poligon

Geo a învățat o metodă de a fixa n puncte pe un cerc de rază r, astfel încât să împartă cercul în n coarde egale ca lungime. Apoi și-a ales un număr k și a început să unească punctele succesiv, din k în k, păstrând același sens, până când a ajuns în punctul din care a pornit. Astfel, dacă a fixat $n = 10$ puncte pe cerc pe care le-a numerotat 1, 2, ..., 10 și și-a ales $k = 6$, atunci el unește punctul 1 cu 7, apoi pe 7 cu 3, apoi 3 cu 9, apoi 9 cu 5, și în sfârșit 5 cu 1.



Apoi a colorat poligonul format în interior, pornind din centrul cercului și fără a depăși vreuna dintre liniile desenate. El se întreabă în final câte laturi are poligonul colorat și care este aria acestuia.

Pentru numerele naturale date n, k și r, se cere numărul de laturi L ale poligonului colorat și aria S a acestuia (cu 2 zecimale exacte).

Date de intrare

Din fișierul **poligon.in** se citesc trei numere naturale n, k și r, despărțite prin câte un spațiu.

Date de ieșire

În fișierul **poligon.out** se vor scrie, pe prima linie numărul L de laturi ale poligonului colorat, iar pe linia a doua numărul real reprezentând aria acestuia.

Restricții și precizări

- $3 < n < 10.001$;
- $0 < k < n$;
- pentru n par, $2 \cdot k \ne n$;
- $10 < r < 501$;
- pentru fiecare test, dacă numărul de laturi determinat este corect, se acordă 20% din punctajul maxim de pe testul respectiv, iar dacă și aria determinată este corectă atunci se acordă punctajul maxim.

Exemple

poligon.in	poligon.out
10 6 100	5
	3468.92

poligon.in	poligon.out
30 13 200	30 5452.04

Timpi de execuție:
- *Microsoft Windows*: 0,1 secunde/test
- *Linux*: 0,1 secunde/test

P040517: Poliție

Gigel și *Costel* sunt doi polițiști cu experiență. Ei lucrează împreună de mulți ani și de multe ori au fost nominalizați pentru premiul "Polițiștii anului". Anul acesta sunt hotărâți să-l câștige și pentru aceasta trebuie să încaseze cât mai mulți bani din amenzi.

În fiecare zi, *Gigel* și *Costel* pot aplica trei tipuri de amenzi:
- amendă de tipul 1, care se acordă pietonilor care traversează neregulamentar și pentru care se încasează suma S_1 și are durata de aplicare T_1;
- amendă de tipul 2, care se acordă șoferilor de autovehicule care încalcă regulile de circulație și pentru care se încasează suma S_2 și are durata de aplicare T_2;
- amendă de tipul 3, care se acordă șoferilor de mașini grele, cu transport ilegal de marfă și pentru care se încasează suma S_3 și are durata de aplicare T_3.

Amenzile de tipul 1 și 2 pot fi aplicate de un singur polițist (*Gigel* sau *Costel*). Pentru o amendă de tipul 3, *Gigel* și *Costel* trebuie să lucreze împreună (unul verifică actele de transport, iar celălalt verifică marfa).

Durata de aplicare a unei amenzi reprezintă timpul necesar polițiștilor pentru a verifica acte, a scrie proces verbal etc. Dacă un polițist aplică o amendă la momentul x, iar durata aplicării amenzii este y, polițistul care aplică amenda va deveni disponibil abia la momentul $x + y$. Polițiștii nu fac minute suplimentare, ei sunt în activitate din minutul 1 până în minutul T exclusiv, deci trebuie să fie liberi să plece acasă în minutul T. Pentru că în noul cod rutier nu le mai este permis polițiștilor să tragă șoferii pe dreapta și să-i lase să aștepte, pentru a aplica o amendă de tipul 1 sau 2 trebuie să fie liber măcar un polițist, iar pentru a aplica o amendă de tipul 3 ambii polițiști trebuie să fie liberi la momentul în care survine evenimentul.

Cei doi polițiști stau la pândă și observă evenimentele din trafic. Dacă nu pot aplica amenzi pentru toate evenimentele care intervin în trafic, ei sunt nevoiți să le aleagă pe acelea care, în total, le aduc mai mulți bani.

Scrieți un program care să determine suma maximă pe care o pot încasa din amenzi *Gigel* și *Costel* într-o tură.

Date de intrare

Fișierul de intrare **politie.in** conține:
- pe prima linie numărul natural T, reprezentând minutul la care cei doi polițiști sunt liberi să plece acasă
- pe linia a doua, două numere naturale S_1 T_1 (suma încasată și durată aplicării unei amenzi de tipul 1);
- pe linia a treia, două numere naturale S_2 T_2 (suma încasată și durată aplicării unei amenzi de tipul 2);
- pe linia a patra, două numere naturale S_3 T_3 (suma încasată și durată aplicării unei amenzi de tipul 3);
- pe linia a cincea, un număr natural N (numărul de evenimente survenite în trafic);
- pe fiecare dintre următoarele N linii se află două numere naturale *tip timp* (*tip* poate fi 1, 2 sau 3 și reprezintă tipul amenzii care poate fi aplicată; *timp* reprezintă timpul la care a survenit evenimentul, exprimat în număr de minute față de începutul turei).

Numerele scrise pe aceeași linie sunt separate între ele prin câte un spațiu. Evenimentele sunt date în ordine cronologică.

Date de ieșire

Fișierul de ieșire **politie.out** conține o singură linie pe care este scrisă suma maximă care poate fi încasată.

Restricții și precizări

- $0 < T, T_1, T_2, T_3 < 201$;
- $0 < S_1, S_2, S_3 < 51$;
- $0 < N < 501$;
- există evenimente care intervin simultan în trafic.

Exemplu

politie.in	politie.out
300 10 20 30 30 50 25 8 1 5 3 10 3 20 1 130 2 142 2 160 3 180 2 280	140

Explicație

Aplică amândoi amenda de tipul 3 din minutul 10, apoi *Costel* aplică singur amenda de tipul 1 din minutul 130, apoi *Gigel* aplică singur amenda de tipul 2 din minutul 142, apoi amândoi aplică amenda de tipul 3 din minutul 180.

Timpi de execuție:
- *Microsoft Windows*: 1,2 secunde/test
- *Linux*: 0,2 secunde/test

P040518: Sea

Pe mare se află N vapoare. Malul este în mod curios perfect drept și este reprezentat prin axa Ox a sistemului de coor-

donate. Cele N vapoare sunt reprezentate prin perechi de coordonate (Vx_i, Vy_i), unde Vy_i este strict pozitiv (marea este deasupra axei Ox). Pe mal se află M faruri, date prin coordonatele lor Fx_i (fiind exact la limita dintre mare și uscat, coordonata y a lor este întotdeauna 0). Cele M faruri sunt ciudate pentru că ele nu pot lumina decât în stânga. Astfel aria luminată de fiecare far i este delimitată de un sfert de cerc cu o rază Fr_i. Mai exact, un vapor este luminat de un anumit far dacă se află în stânga farului (are coordonata x mai mică) și distanța de la far la vapor este mai mică sau egală cu valoarea Fr_i asociată farului respectiv.

Pentru fiecare far se mai dă și un număr natural strict pozitiv Fn_i. Din motive greu de înțeles, șeful portului dorește ca fiecare far i să lumineze cel puțin Fn_i vapoare (un vapor poate fi luminat de mai multe faruri). El dorește consum minim de energie și vrea să afle pentru fiecare far raza minimă necesară pentru a lumina numărul cerut de vapoare.

Determinați pentru fiecare far valoarea Fr_i care reprezintă raza minimă necesară pentru ca farul să lumineze cel puțin Fn_i vapoare.

Date de intrare

Prima linie a fișierului **sea.in** conține două numere întregi N și M, separate printr-un spațiu, reprezentând numărul de vapoare, respectiv numărul de faruri.

Fiecare dintre următoarele N linii conține câte o pereche de numere reale separate printr-un spațiu Vx_i și Vy_i (coordonatele vapoarelor).

Fiecare dintre următoarele M linii conține câte o pereche de numere separate printr-un spațiu, unul real Fx_i și unul întreg Fn_i (coordonatele orizontale și numerele asociate farurilor).

Date de ieșire

Fișierul **sea.out** va conține M linii, fiecare linie conținând un număr real, dat cu patru zecimale: pe linia i se află raza minimă necesară pentru ca farul i să lumineze Fn_i vapoare.

Restricții

- $1 \le N \le 400$, $1 \le M \le 100.000$;
- $0 < y, r < 100.000$;
- $-100.000 < x < 100.000$;
- $1 \le Fn_i \le N$;
- în fișierul de intrare farurile sunt sortate crescător după coordonatele x;
- nu vor exista două vapoare, sau un far și un vapor cu aceeași valoare pentru coordonata x;
- pot exista două sau mai multe faruri cu aceeași coordonată x, caz în care ele vor fi unul lângă altul în fișierul de intrare (din moment ce sunt sortate după x);
- ordinea în care apar în fișierul de intrare farurile având aceeași coordonată x nu este definită;
- pot exista chiar două faruri identice.
- numerele reale din fișierul de intrare vor avea cel mult 4 zecimale exacte;
- rezultatul va fi verificat cu o precizie de 0,001 (rezultatul va fi considerat corect dacă modulul diferenței dintre rezultatul corect și cel furnizat de concurent nu depășește 0,001);
- există întotdeauna soluție (pentru fiecare far i vor exista întotdeauna cel puțin Fn_i vapoare în stânga lui).

Exemplu

sea.in	sea.out
3 5	5.0990
-0.5 0.5	0.7071
-2 5	5.3852
3 4	0.7071
-1 1	4.4721
0 1	
0 2	
0 1	
5 1	

Timpi de execuție:

- *Microsoft Windows*: 1,6 secunde/test
- *Linux*: 0,8 secunde/test

Câștigătorii

Clasa a V-a

Cătălin Bălan, Cluj - premiul I
Vlad Tătăranu, București - premiul I
Ioana Pandele, București - premiul III

Clasa a VI-a

Petre Clinciu-Glișca, Gorj - premiul I
Ada-Mihaela Solcan, București - premiul I
Otilia Stretcu, Mehedinți - premiul I

Clasa a VII-a

Liviu Mirea-Ghiban, București - premiul I
Alexandru-Petrișor Pajarcu, Dolj - premiul I

Clasa a VIII-a

Casian-Valentin Andrei, Argeș - premiul I
Dan Grigoriu, București - premiul II
Andrei Ștefan, București - premiul II

Membrii Lotului Național de Informatică:

Cristian Botău, Bihor
Mihai Ciucu, Dolj
Dan-Leonard Crestez, Brăila
Mircea Digulescu, București
Marius Dumitran, București
Constantin Dumitrașcu, Vrancea
Dan-Ionuț Fechete, Suceava
Silviu Gânceanu, București
Dan Ghinea, București
Cătălin-Paul Grigoraș, Suceava
Emilian Miron, Bacău
Alexandru Moșoi, București
Marius Nicolae, Olt
Daniel Păsăilă, Suceava
Mircea Pașoi, Pralnova
Ionuț Prunache, Vrancea
Dan Spătărel, București
Sorin Stancu-Mara, București
George-Cristian Strat, Vaslui
Adrian Vlădu, București

Craiova, 26-28 martie 2004

Concursul de programare ȘTEFAN ODOBLEJA

În perioada 26-28 martie 2004 a avut loc la Craiova ediția a treia a Concursului Interjudețean de Programare "Ștefan Odobleja". Datorită unor probleme organizatorice concursul pe echipe nu a mai avut loc, concurenții având posibilitatea să-și arate cunostințele de algoritmică și să-și dovedească abilitățile de programatori în cadrul concursului individual.

Clasa a IX-a

P040519: Loto

Văzând *Aurel* cum toată lumea se îmbulzește să joace la loto 6 din 49, se hotărăște să își încerce și el norocul. Un vecin îi explică regulile acestui joc. Există un bilet care trebuie completat. Pe el se află un careu cu numere de la 1 la 49, deci careul conține 49 de pătrățele numerotate. Dacă el marchează 6 numere, se consideră că a ales o variantă; aceasta costă 30 000 de lei. Jocul constă în extragerea a 6 numere. Dacă toate numerele extrase se regăsesc printre cele marcate de el în careu, atunci biletul este câștigător.

Dacă el marchează M numere ($M > 6$), se consideră că a ales toate variantele de câte 6 numere dintre cele M completate. Deci, dacă va marca 7 numere, va avea 7 variante posibile, fiecare variantă reprezentând câte 6 numere din cele 7.

De exemplu, dacă *Aurel* marchează primele numere de la 1 la 7, următoarele variante se obțin:

```
1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 6 7
1 2 3 5 6 7
1 2 4 5 6 7
1 3 4 5 6 7
2 3 4 5 6 7
```

Dacă va marca 8 numere, va avea 28 de variante de câte 6 numere care se pot crea. Costul unui bilet este egal cu numărul de variante care se pot forma înmulțite cu costul unei variante.

Aurel are o sumă de N lei. Ajutați-l să cumpere bilete de loto astfel: pe primul bilet să marcheze cât mai multe numere, dar costul biletului să nu depășească suma de N lei. Din suma de bani eventual rămasă trebuie să cumpăre alte bilete, respectând același principiu.

Date de intrare

Fișierul `LOTO.IN` conține un singur număr natural N care reprezintă suma de bani pe care vrea *Aurel* să o joace la loto.

Date de ieșire

Fișierul de ieșire `LOTO.OUT` trebuie să conțină pe fiecare linie câte două numere.

Primul număr de pe o linie reprezintă numărul de numere marcate M_1 pe un bilet, iar al doilea număr reprezintă numărul de bilete cu M_1 numere marcate. Următoarea linie, dacă există, are numărul M_2 ($M_2 < M_1$) și numărul de bilete în care se pot marca cele M_2 numere ș.a.m.d.

Restricție

- $N \leq 15\,000\,001$.

Exemplu

`LOTO.IN`

```
1300000
```

`LOTO.OUT`

```
8 1
7 2
6 1
```

Explicație

Cu suma de 1.300.000 lei, *Aurel* poate cumpăra un bilet cu 8 numere marcate, două bilete cu 7 numere marcate și un bilet cu 6 numere marcate.

Timp de execuție: 1 secundă/test

P040520: Premii

Compania C dorește să premieze o parte dintre clienții săi la sfârșitul zilei. Numărul premiilor fiind mic, se stabilește următoarea strategie:

- toți clienții se vor înscrie pe o listă, fiecare primind un număr de ordine; primul client va primi numărul de ordine 2, al doilea client va primi numărul de ordine 3, al treilea client va primi numărul de ordine 4 ș.a.m.d.;
- premiile se vor acorda astfel: primul client, cu numărul de ordine 2, este declarat câștigător, iar clienții cu numerele de ordine 4, 6, 8, 10 etc. sunt necâștigători și sunt excluși din listă; următorul client din listă, cel cu numărul de ordine 3, este declarat câștigător, iar clienții cu numerele de ordine 9, 15, 21, 27 etc. sunt necâștigători și sunt excluși din listă; următorul client din listă, cel cu numărul de ordine 5, este declarat câștigător, iar clienții cu numerele de ordine 19, 35, 49, 65 etc. sunt necâștigători și sunt excluși din listă;
- procedeul continuă până când nu mai rămâne nici un client în listă.

Clienții câștigători vor fi cei cu numerele de ordine: 2, 3, 5, 7, 11, 13, 17 etc.

Dându-se un număr I, să se determine numărul de ordine al celui de al I-lea client câștigător.

Date de intrare

Fișierul de intrare **C.IN** conține pe prima linie un număr întreg pozitiv N care reprezintă numărul de linii care se află în continuare. Fiecare dintre următoarele N linii, va conține un număr intreg I care reprezintă un câștigător.

Date de ieșire

Fișierul de ieșire **C.OUT** trebuie să conțină N linii. Fiecare linie a fișierului de ieșire va trebui să conțină un singur număr care reprezintă numărul de ordine al unui câștigător corespunzător unei linii din fișierul de intrare.

Datele din fișierul de ieșire trebuie scrise în ordinea celor din fișierul de intrare.

Restricție

- $1 \le I \le 3000$.

Exemplu

C.IN

```
5
1
2
3
20
73
```

C.OUT

```
2
3
5
83
437
```

Timp de execuție: 1 secundă/test

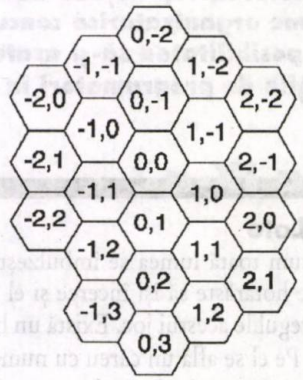
Clasa a X-a

P040521: Albinuța Zora

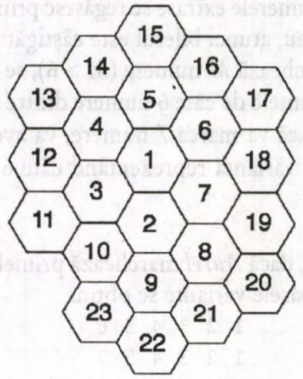
Zora este o albinuță. Ea trăiește într-un stup împreuna cu alte albine. Acest stup este alcătuit din mai multe celule hexagonale în care se stochează miere.

Zora are o problemă. *Somnorilă* i-a spus unde poate să-l întâlnească, dar, pentru că *Somnorilă* este un bondar, iar *Zora* este o albină lucratoare, ei utilizează sisteme de coordonate diferite.

Zora care, de regulă, zboară direct către o celulă anume, și-a dezvoltat un sistem de coordonate bidimensional peste întregul stup, iar celula din mijlocul stupului are coordonatele (0, 0).



Somnorilă, mult mai leneș, este obișnuit să meargă de jur împrejur, numărând celulele în sensul acelor de ceasornic, începând cu 1 în mijlocul stupului.



Ajutați-o pe *Zora* sa obțină conversia din sistemul lui *Somnorilă* în sistemul ei.

Date de intrare

Fișierul de intrare **ZORA.IN** conține unul sau mai multe numere întregi reprezentând numerele lui *Somnorilă*. Fiecare număr este situat pe câte o linie și este urmat de marcajul de sfârșit de linie (<ENTER>).

Ultima linie din fișier conține valoarea 0. Nu trebuie produsă nici o ieșire pentru 0.

Date de ieșire

Fișierul de ieșire **ZORA.OUT** trebuie să conțină coordonatele sistemului lui *Zora* corespunzătoare numerelor lui *Somnorilă*. Fiecare pereche de coordonate se va scrie pe câte o linie

și este urmată de marcajul de sfârșit de linie (<ENTER>), iar cele două coordonate corespunzătoare unei perechi trebuie separate între ele printr-un spațiu.

Restricție

- numărul de celule din stup este mai mic decât 100000.

Exemplu

```
ZORA.IN
1
2
3
4
5
0

ZORA.OUT
0 0
0 1
-1 1
-1 0
0 -1
```

Timpi de execuție: 1 secundă/test

P040522: Dreptunghiuri

Pe ecranul calculatorului sunt dispuse dreptunghiuri care au laturile paralele cu axele de coordonate (marginile ecranului).

Sarcina voastră este să scrieți un program care acceptă la intrare astfel de dreptunghiuri, date prin coordonatele pe ecran a două vârfuri opuse. Programul va trebui să determine perimetrul și aria reuniunii de dreptunghiuri de pe ecran.

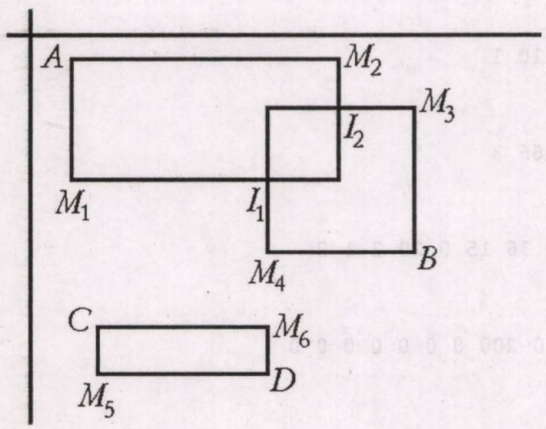


Figura 1: Exemplu de dreptunghiuri. Reuniunea lor este: $(A, M_1, I_1, M_4, B, M_3, I_2, M_2) \cup (C, M_5, D, M_6)$

Date de intrare

Fișierul de intrare **DREPT.IN** va conține pe fiecare linie coordonatele de pe ecran (numere întregi) a două vârfuri opuse ale unui dreptunghi cu laturile paralele cu axele de coordonate.

Dacă două vârfuri opuse ale unui dreptungi sunt $M_1(x_1, y_1)$ și $M_2(x_2, y_2)$, atunci o linie din fișierul de intrare va conține numerele x_1, y_1, x_2 și y_2, separate între ele printr-un singur spațiu.

Date de ieșire

Fișierul de ieșire **DREPT.OUT** va conține pe prima linie perimetrul, iar pe cea de-a doua linie aria reuniunii de dreptunghiuri.

Restricții și precizări

- numărul maxim de dreptunghiuri din fișierul de intrare este 200;
- în fișierul de intrare nu există două dreptunghiuri suprapuse în totalitate.

Exemplu

```
DREPT.IN                DREPT.OUT
0 20 100 100            680
60 0 120 90             21300
190 10 110 110
150 50 20 120
```

Timpi de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P040523: Rumbu

Gigel se duce în țara *Rumbu*, undeva pe planeta *Ajelbodo*, și nu cunoaște limba băștinașilor.

După mai multe peripeții reușește să fure un papirus în care sunt descrise regulile de formare a cuvintelor din limba *rumbu*. Cuvintele în limba *rumbu* sunt formate numai din literele mici ale alfabetului latin $\Sigma = \{'a', \ldots, 'z'\}$.

Un cuvânt se construiește în urma aplicării unor reguli de forma: $M_1 \to M_2M_3$ sau $M_4 \to \alpha$, unde M_1, M_2, M_3 și M_4 sunt metasimboluri oarecare (nu aparțin alfabetului Σ al limbii), iar $\alpha \in \Sigma$. Mulțimea metasimbolurilor este formată din litere mari din mulțimea {'A', ..., 'Z'}.

Orice cuvânt aparținând limbii este obținut pornind de la o regulă care are în partea stângă metasimbolul *S*. Un cuvânt este considerat corect dacă:

- s-a obținut prin aplicarea repetată a regulilor;
- nu conține nici un metasimbol.

Dacă pe papirus se află următoarele reguli:

R1) $S \to AB$
R2) $A \to a$
R3) $B \to b$
R4) $A \to AA$

atunci cuvântul '*aaab*' este corect deoarece se poate obține astfel:

$$S \xrightarrow{R_1} AB \xrightarrow{R_4} AAB \xrightarrow{R_2} aAB \xrightarrow{R_4} aAAB \xrightarrow{R_3} aAAb \xrightarrow{R_2} aaAb \xrightarrow{R_2} aaab$$

Ajutați-l pe *Gigel* să dezlege misterele limbii *rumbu*.

Dându-se un set de reguli de pe un papirus și mai multe șiruri de caractere, verificați dacă acestea aparțin limbajului *rumbu*.

Date de intrare

Pe prima linie a fișierului de intrare **PROB1.IN** se află un număr natural N care reprezintă numărul regulilor din mulțimea P (de pe papirus). Pe următoarele N linii se află regulile, câte una pe linie. O regulă poate avea forma:

A->BC sau A->a.

Ultima linie a fișierului de intrare va conține un cuvânt, format numai din simboluri din Σ, pentru care trebuie determinată apartenența la limbajul rumbu.

Date de ieșire

Fișierul de ieșire **PROB1.OUT** va conține pe o linie răspunsul DA în cazul în care cuvântul din fișierul de intrare aparține limbajului *rumbu* și NU în caz contrar.

Restricție

- $N \leq 200$;
- numărul de caractere ale unui cuvânt pentru care trebuie determinată apartenența la limbajul *rumbu* este cel mult egal cu 250.

Exemplu

PROB1.IN

```
2
S->SS
S->s
ssss
```

PROB1.OUT

```
DA
```

Timpi de execuție: 3 secunde/test

P040524: Multiprocesor

O companie a achiziționat un calculator multiprocesor care dispune de N procesoare. Aceste procesoare sunt așezate liniar și sunt etichetate cu numere cuprinse între 1 și N. Ele sunt utilizate pentru a procesa sarcini independente. Fiecare sarcină nou sosită este distribuită, în mod arbitrar, unuia dintre cele N procesoare. Uneori este posibil ca un procesor să aibă atașate prea multe sarcini, iar alte procesoare să aibă prea puține (posibil chiar nici una). În această situație calculatorul realizează o realocare a sarcinilor procesoarelor pentru a îmbunătăți performanțele.

Procesul de realocare se realizează în runde. În fiecare rundă, fiecare procesor poate transfera cel mult o sarcină fiecăruia dintre procesoarele vecine lui. Vecinii unui procesor i sunt procesoarele $i - 1$ și $i + 1$ (procesoarele 1 și N au numai câte un vecin, pe 2, respectiv pe $N - 1$). Scopul acestei realocări este ca toate procesoarele să aibă de executat același număr de sarcini.

Dându-se un număr de procesoare și numărul de sarcini distribuite fiecărui procesor se cere să se determine numărul minim de runde necesar pentru a se ajunge la starea în care fiecare procesor să aibă de executat același număr de sarcini.

Date de intrare

Fișierul de intrare **PROB2.IN** conține mai multe seturi de date. Fiecare set începe cu o linie care conține un singur număr N - numărul de procesoare. În continuare, avem N numere separate prin spații și/sau caractere de sfârșit de linie, reprezentând numărul de sarcini distribuite fiecărui procesor. Seturile de date sunt separate printr-o linie vidă. Ultimul set de date este urmat de numărul -1 pe o singură linie (număr care nu va fi procesat).

Date de ieșire

Fișierul de ieșire **PROB2.OUT** va conține numărul minim de runde necesare pentru a echilibra sistemul pentru fiecare set în parte. Dacă pentru un set nu este posibil ca fiecare procesor să aiba același număr de sarcini de executat se va afișa ca rezultat -1.

Restricții și precizări

- $1 \leq N, M \leq 10000$;
- numărul de sarcini atașate fiecărui procesor este cuprins între 0 și 32767, iar numărul total de runde nu va depăși un *longint*.

Exemplu

PROB2.IN

```
2
49 50

3
1 10 1

3
0 66 3

8
16 16 15 0 20 2 1 2

10
0 0 200 0 0 0 0 0 0 0

-1
```

PROB2.OUT

```
-1
3
23
22
140
```

Timpi de execuție: 1 secundă/test

Teoria jocurilor: numerele Sprague Grundy

Cosmin-Silvestru Negruşeri

Domeniu relativ nou şi încă necercetat în adâncime, teoria jocurilor este o ramură a matematicii în care de multe ori primează inventivitatea şi nu cunoştinţele. Tocmai din acest motiv, în cadrul acestui articol vom introduce câteva noţiuni teoretice care ne vor ajuta în rezolvarea unor probleme din teoria jocurilor.

Ingeniozitatea celor pasionaţi poate fi testată prin introducerea unor probleme de teoria jocurilor la concursurile de matematică şi informatică.

Deoarece în *România*, *teoria jocurilor* nu este studiată în şcoli, problemele din acest domeniu pot pune în dificultate concurenţii.

NIM

Pentru început ne vom familiariza cu jocul clasic *NIM*:

Se consideră n grămezi de pietre. Doi jucători, vor ridica (alternativ) oricâte pietre dintr-o singură grămadă. Câştigătorul este cel care ia ultima piatră.

Pentru cazul trivial în care numărul de grămezi este egal cu 1, primul jucator are evident strategie de câştig, el putând lua toate pietrele din grămadă.

Dacă numărul de grămezi este egal cu 2, primul jucător are strategie de câştig atunci când numărul de pietre din prima grămadă este diferit de numărul de pietre din cea de a doua, strategia lui fiind cea de a aduce tot timpul grămada mai mare la numărul de pietre al grămezii mai mici, şi cum jocul este finit, înseamnă că primul jucător o să aducă jocul în starea (0, 0).

Dacă numărul de grămezi este mai mare decât doi strategia se complică şi nu se mai observă cu "ochiul liber". Stările câştigătoare pentru mai multe grămezi sunt acele stări pentru care suma *XOR* a numerelor de pietre din grămezi este diferită de 0, restul stărilor fiind de pierdere.

De exemplu, dacă avem o gramadă cu o piatră, o gramadă cu trei pietre, o gramadă cu cinci pietre şi o gramadă cu şapte pietre:

o
ooo
ooooo
ooooooo

atunci vom avea:

$1 = (0001)_2$
$3 = (0011)_2$
$5 = (0101)_2$
$7 = (0111)_2$

efectuând *XOR* (operatorul ^ în *C/C++*) între reprezentările binare ale numerelor obţinem $0 = (0000)_2$.

Conform propoziţiei de mai sus această stare este de pierdere.

Să demonstrăm cele afirmate.

Dintr-o poziţie cu suma *XOR* egală cu 0, pentru orice mutare ajungem evident la o poziţie cu suma *XOR* diferită de 0, pentru că luând dintr-o grămadă un număr x de pietre, în suma *XOR* corespunzătoare noii stări bitul cel mai semnificativ al lui x va avea valoarea 1.

Mai rămâne de demonstrat că din orice poziţie cu suma *XOR* diferită de 0 putem trece printr-o mutare convenabilă într-o poziţie cu suma *XOR* egală cu 0.

Căutăm o grămadă care are un număr de pietre mai mare sau egal cu cea mai mare putere a lui 2 care apare în suma *XOR*. Fie x valoarea sumei *XOR* a tuturor grămezilor şi y numărul de pietre din grămada găsită mai devreme. O mutare "câştigătoare" este extragerea din grămada găsită care are y pietre a $y - (x \text{ XOR } y)$ pietre ($x \text{ XOR } y$ este mai mic

decât y pentru că se anulează biții cei mai semnificativi ai lui y și x). Atunci noua sumă *XOR* va fi egală cu 0.

De exemplu: $4 \text{ ^ } 8 \text{ ^ } 17 = (00100)_2 \text{ ^ } (01000)_2 \text{ ^ } (10001)_2 = (11101)_2 = 29$

Mutarea câștigătoare constă în a lua din cea de-a treia gramadă un număr de pietre egal cu:

$17 - (17 \text{ ^ } 29) = 17 - 17 \text{ ^ } 29 = 5 = (00101)_2$.

După acest pas grămezile vor avea 4, 8, 12 pietre. Ne aflăm astfel într-o stare cu suma *XOR* egală cu 0.

Exemplificăm în continuare câteva probleme în care se folosește strategia de la jocul *NIM*.

Problema 1

Pe o tablă de șah, care are $n \cdot m$ căsuțe, sunt plasați pe prima linie n pioni albi și pe ultima linie n pioni negri. Fiecare dintre cei doi jucători poate muta un singur pion, care îi aparține, un număr strict pozitiv de căsuțe în sus sau în jos, astfel încât să nu ajungă vreun pion alb să fie mai jos decât pionul negru de pe aceeași coloană. Pierde jucătorul care nu mai poate muta.

Această problemă este o deghizare a jocului *NIM*, numărul de pătrățele libere între pionul alb și pionul negru de pe coloana i putând fi considerat numărul de pietre din grămada i.

Singura diferență este că se pot adăuga pietre la grămadă (existând posibilitatea mutării înapoi).

Această problemă se rezolvă ușor, jucătorul care are strategie de câștig putând evita asemenea mutări. O astfel de mutare poate fi utilă numai pentru jucătorul care este într-o poziție de pierdere.

Când jucatorul care nu are strategie de câștig mută înapoi x casuțe, celălalt jucător va muta pionul propriu de pe aceeași coloană cu x căsuțe în față, astfel ajungându-se la aceeași stare cu cea existentă cu două mutări anterior (considerând diferența pozițiilor).

Problema 2

Această problemă a fost propusă spre rezolvare participanților la barajul pentru lărgirea lotului național din 1997.

Pe o linie sunt plasate la coordonate întregi $2 \cdot n$ piese roșii și albastre.

Fiecare piesă roșie poate fi mutată în dreapta oricâte poziții astfel încât să nu sară peste o piesă albastră, iar piesele albastre pot fi mutate oricâte poziții la stânga astfel încât să nu depășească vreo piesă roșie. Piesele vor alterna: roșu, albastru, roșu, albastru etc. Pierde jucătorul care nu mai poate muta.

Această problemă poate fi, de asemenea, redusă la jocul *NIM*. Diferențele pozițiilor perechilor de piese roșii și albastre consecutive constituie numărul de pietre al grămezilor din jocul *NIM*.

Problema 3

(*El Judge MIPT online programming contest Nim Game Give Away!*)

Se consideră n grămezi de pietre, jucătorii mută alternativ, fiecare jucător extrăgând oricâte pietre dintr-o singură grămadă. Cel care ia ultima piatră pierde jocul.

Strategia acestui joc este similară cu cea aplicată în jocul *NIM* cu câteva mici diferențe.

Jucătorul care are strategie de câștig în poziția curentă în cadrul jocului *NIM* face aceeași mutare pe care ar face-o în cazul jocului *NIM*, exceptând cazul în care această mutare lasă doar grămezi cu o singură piatră și numărul acestor grămezi este par. În această situație, dacă ar trebui să ia x pietre, jucătorul poate lua x - 1 pietre din grămada actuală, pentru ca numărul de grămezi să fie impar și el să facă ultima mutare.

Numerele Sprague Grundy

Jocurile care prezintă interes pentru jucători sunt acelea care necesită examinarea unui număr foarte mare de stări pentru determinarea strategiei de câștig, deoarece în caz contrar câștigătorul s-ar cunoaște chiar de la început. Spre deosebire de aceștia, matematicienii sau informaticienii sunt interesați de determinarea unor strategii pentru astfel de jocuri.

Toate jocurile impartiale cu doi jucători cu informație totală pot fi reduse la jocul *NIM* care se joacă cu niște grămezi virtuale, în care mutările posibile sunt extragerea oricâtor pietre dintr-o grămadă sau adăugarea oricâtor pietre la o grămadă (așa cum am menționat anterior, adăugarea de pietre la o grămadă nu complică analiza jocului).

Afirmația anterioară constituie un rezultat al *Teoriei Sprague-Grundy*. ***Roland Percival Sprague*** (1936) și ***Patrick Michael Grundy*** (1939) sunt doi matematicieni care s-au ocupat, independent, de teoria jocurilor impartiale.

Majoritatea jocurilor impartiale se pot reduce la jocul prezentat în problema *Pioni* de la runda 47 a concursului de programare *Bursele Agora*. Acolo se menționa următorul joc: *Se consideră un graf aciclic care conține în noduri câțiva pioni, jucătorii alternează la mutare și fiecare poate muta câte un pion pe un arc care iese din nodul în care este situat pionul. Pierde jucătorul care nu poate muta.*

Cum graful este aciclic, jocul este finit și are întotdeauna un câștigător. Practic, acest joc este suma unor jocuri, mai precis suma a mai multor jocuri cu un singur pion pe graful aciclic.

Jocul cu un singur pion poate fi analizat destul de ușor, fiecare nod al grafului putând fi colorat cu alb sau negru după cum există sau nu strategie de câștig dacă în nodul curent ar fi poziționat pionul. Această colorare poate fi realizată ușor dacă se pornește de la nodurile fără urmaș și la fiecare pas se colorează câte un nod ai cărui urmași sunt deja colorați.

Orice joc imparțial poate fi redus la un joc cu un singur pion. Nodurile sunt pozițiile jocului și arcele grafului sunt mutările posibile din fiecare poziție. Jocul inițial poate fi și el transformat, dar numărul de noduri crește foarte mult (pentru n pioni și m noduri, numărul de noduri din jocul transformat este n^m) și nu este practic să colorăm graful rezultat. Folosind teoria dezvoltată de *Sprague* și *Grundy*, putem reduce complexitatea analizei jocului cu n pioni la complexitatea analizei jocului cu un pion.

Vom introduce funcția *mex* cu semnificația: $mex(S)$ este elementul minimal natural care nu aparține mulțimii S. Pentru fiecare nod x al grafului aciclic considerat, vom calcula valoarea funcției $gx = mex(gx_1, gx_2, ..., gx_k)$, unde x_1, x_2, ..., x_k sunt urmașii nodului x în graf. Pentru nodurile fără urmași $gx = 0$.

Analog jocului cu un singur pion, graful poate fi etichetat din aproape în aproape, pornind de la nodurile fără urmași.

Observăm că, dacă $gx = 0$ în jocul cu un singur pion situat în nodul x nu avem strategie de câștig, iar dacă gx este diferit de 0 avem strategie de câștig. Restul informației ne ajută la determinarea unei strategii pentru jocul cu n pioni.

Observăm că putem reduce acest joc la jocul *NIM* în care grămada virtuală asociată pionului i are un număr de pietre egal cu valoarea g a nodului în care este situat pionul i.

De ce este acest joc echivalent cu jocul NIM?

Așa cum la *NIM* puteam lua oricâte pietre dintr-o grămadă, aici putem muta pionul dintr-un nod cu valoarea g într-un nod astfel încât noua valoare pentru pionul i poate fi orice număr de la 0 la $g - 1$. Prin urmare, pentru a verifica dacă suntem într-o poziție de câștig în jocul cu pionii, putem aplica strategia jocului *NIM* și obținem că suntem într-o poziție de câștig dacă suma *XOR* a numerelor din nodurile ocupate de cei n pioni este diferită de 0. Această sumă se numește funcția *Sprague Grundy*, $SG(i_1, ..., i_n) = gi_1 \wedge gi_2 \wedge gi_3 \wedge ... \wedge gi_n$.

Problema de la runda 47 se rezolvă acum ușor efectuând o sortare topologică a nodurilor grafului aciclic și numerotând nodurile folosind funcția *mex*.

Să studiem acum alte probleme care se pot rezolva folosind numerele *Sprague Grundy*.

Problema 4

Problema *Joc* a rundei 13 a concursului de programare *Bursele Agora* putea fi rezolvată în acest mod.

În acea problemă se cerea să verificăm existența unei strategii de câștig pentru un joc similar cu *NIM* în care se putea lua dintr-o grămadă o piatră sau un număr prim de pietre.

Dacă determinăm valorile *Sprague Grundy* pentru grămezi de dimensiuni mici putem observa că se repeta o succesiune de numere: 0 1 2 3 0 1 2 3 ...

Putem demonstra prin inducție că această secvență se va repeta la nesfârșit.

Pentru o grămadă de dimensiune n valoarea asociată va fi n modulo 4.

Pentru $0 \le n \le 3$ afirmația este adevărată.

Vom presupune afirmația adevărată pentru toate valorile $m < n$. Să demonstrăm acum că este adevărată și pentru n.

Deoarece putem lua din n pietre una, două sau trei pietre, mai rămâne valoarea n modulo 4 care nu este eliminată încă din valorile potențiale asociate grămezii de dimensiune n.

Vom arăta în continuare că această valoare nici nu va fi eliminată. Eliminarea ei ar însemna că putem lua din n un număr p de pietre și atunci din $(n - p)$ modulo 4 = n modulo 4, am avea: p modulo 4 = 0, dar p este un număr prim, deci valoarea *Sprague Grundy* asociată unei grămezi de dimensiune n este n modulo 4.

Problema 5

(*El Judge MIPT online programming contest, Stone game*)

Se consideră k grămezi cu $n_1, n_2, ..., n_k$ pietre fiecare. Când este rândul său, un jucător poate lua dintr-o gramadă 2^m pietre. Jucătorul care ia ultima piatră câștigă.

Restricții: $k \le 50$, $n_i \le 10^{200}$.

Să determinăm valorile *Sprague Grundy* pentru grămezi mici:

0 : 0; 1 : 1; 2 : 2; 3 : 0; 4 : 1; 5 : 2; 6 : 0

Observăm și aici secvența repetitivă 0 1 2 0 1 2, deci am putea trage concluzia că valoarea *Sprague Grundy* asociată unei grămezi de dimensiune n este n modulo 3. Această afirmație este adevarată și urmează aceeași demonstrație ca în cazul problemei anterioare, iar restul modulo 3 pentru un număr cu 200 de cifre este simplu de găsit determinând suma cifrelor numărului.

Problema 6

(*El Judge MIPT online programming contest, Stone game II*)

Se consideră k grămezi de pietre cu $n_1, n_2, ..., n_k$ pietre. Un jucător poate lua dintr-o grămadă la mutarea lui un număr pozitiv de pietre dar nu mai mult de jumătate din pietrele din grămadă. Jucătorul care nu mai poate muta pierde.

Restricții: $k \le 50$, $n_i \le 100000$.

Numărul 100000 nu este foarte mare și valorile *Sprague Grundy* pot fi determinate *off-line* și incluse în programul nostru ca și constante.

Putem scrie pentru valori mici secvența *Sprague Grundy*:

1 2 3 4 5 6 7 8 9 10 11 12 13 14
0 1 0 2 1 3 0 4 2 5 1 6 3 7

Pentru n impar valoarea asociată este aceeași cu valoarea asociată lui $n/2$, și pentru n par valoarea asociată este $n/2$. Acest lucru se poate demonstra prin inducție matematică.

Problema 7

(CEOI 2000, Cluj-Napoca, problema Sticks)

Se consideră n ($n \leq 10$) rânduri de bețe pe o masă, cu S_i ($S_i \leq 10$) bețe aliniate pe fiecare rând și doi jucători. Bețele de pe rândul i sunt numerotate secvențial de la 1 la S_i.

Cei doi jucători mută alternativ. Fiecare mișcare constă în eliminarea a unu, două sau trei bețe de pe același rând. Bețele trebuie să fie numerotate secvențial, adică să fie consecutive.

De exemplu, un rând are 10 bețe și primul jucător elimină bețele 4, 5, 6, deci vor rămâne numai bețele 1, 2, 3, 4, 8, 9, 10. Al doilea jucător poate lua la rândul său bețele 1, 2, 3, dar nu bețele 3, 7, 8 pentru că acestea nu sunt numerotate consecutiv (bineînțeles că există și alte mutări valide). Câștigă jucătorul care ia ultimul băț de pe masă.

Problema generală are o soluție ingenioasă care ține seama de paritățile rândurilor, dar la această problemă, datorită mărginirii lui S_i ($S_i \leq 10$) nu este necesar să fim ingenioși. Restricția $S_i \leq 10$, ne ajută prin faptul că numărul total de poziții (dacă jucăm pe o singură gramadă), este 2^{10}. Vom reprezenta o poziție printr-un întreg, iar dacă acel întreg are în codificarea lui binară pe poziția i un bit de 1 înseamnă că el reprezintă un rând de bețe care conține în el bățul numerotat cu i. Este ușor de realizat un graf aciclic al mișcărilor pentru un rând (graful este aciclic pentru că la fiecare mutare luăm bețe din configurație). Numerotăm fiecare poziție cu numerele *Sprague Grundy*, și acum problema deciderii dacă suntem sau nu într-o poziție câștigătoare devine banală. În problema inițială trebuia să jucăm împotriva calculatorului și să câștigăm. Putem realiza aceasta folosind mutarea câștigătoare prezentată la jocul *NIM*.

Problema 8

Această problemă a fost propusă spre rezolvare la concursul *Internet Problem Solving Contest* (cel mai prestigios concurs *online*) și o puteți găsi la adresa **http://ipsc.ksp.sk/xxproblems/ipsc2003/g.php**. Ea a fost folosită și la concursul organizat de *.campion* la o rundă *online*. Rezolvarea ei, complicată, folosind numerele *Sprague Grundy* prezentate în acest articol, se poate găsi în [3], pe *site*-ul [7], sau pe *site*-ul concursului *.campion*.

Problema 9

Această problemă a fost propusă spre rezolvare la ediția din acest an a *Olimpiadei Naționale de Informatică* din *Ucraina*.

Doi participanți mănâncă alternant din niște tablete de ciocolată după următoarele reguli:

- taie o tabletă în două, tăietura trebuie să fie paralelă cu una din laturile tabletei și trebuie să nu taie pătrățelele de ciocolată;
- poate să rupă și să mănânce orice linie sau coloană de pătrățele care nu se află pe marginea tabletei;
- poate să rupă și să mănânce toate patrățelele de pe marginea tabletei, cu condiția ca tableta rămasă să aibă cel puțin dimensiunea 1×1.

Nici una dintre aceste trei mutări nu poate fi efectuată asupra unei tablete de dimensiune 1×1. Pierde jucătorul care nu mai poate efectua nici o mutare.

În fișierul de intrare se va afla numărul N ($1 \leq N \leq 100$) de tablete, iar pe următoarea linie sunt N perechi de numere întregi care reprezintă dimensiunile tabletelor.

În fișierul de ieșire se va afla un singur număr întreg reprezentând numărul mutărilor câștigătoare pentru primul jucător.

Pentru această problemă vom calcula matricea $SG_{i,j}$ care reprezintă valoarea *Sprague-Grundy* asociată unei tablete de dimensiuni (i, j).

Să vedem care este recurența care va satisface elementele matricei SG:

$$SG_{i,j} = \text{mex}(SG_{i,k} \wedge SG_{i,j-k}, \quad (1 \leq k < j) \quad \text{mutarea întâi}$$
$$SG_{k,j} \wedge SG_{i-k,j}, \quad (1 \leq k < i)$$
$$SG_{i,k} \wedge SG_{i,j-k-1}, \quad (1 \leq k < j-1) \quad \text{mutarea a doua}$$
$$SG_{k,j} \wedge SG_{i-k-1,j}, \quad (1 \leq k < i-1)$$
$$SG_{i-2,j-2}) \quad (i > 2 \text{ și } j > 2) \quad \text{mutarea a treia}$$

Acum, pentru a calcula numărul de mutări câștigătoare efectuăm asupra fiecărei tablete din fișierul de intrare toate mutările posibile care sunt cel mult de $4 \cdot 100 + 1$ și facem suma *XOR* a valorilor *Sprague Grundy* pentru restul tabletelor neimplicate în mutare și a tabletelor rezultate din mutare. Pentru a calcula $SG_{i,j}$ trebuie să parcurgem cel mult $2 \cdot i + 2 \cdot j + 1$ valori obținute. Astfel, algoritmul de determinare al valorilor matricei SG are ordinul de complexitatea $O(N^3)$.

Complexitatea algoritmului care determină numărul de mutări câștigătoare este $O(N^2)$.

Am văzut că aceste numere sunt folositoare pentru rezolvarea unor probleme de jocuri combinatorice. Chiar dacă numărul stărilor grafului nostru aciclic poate să fie foarte mare, putem să ne dăm seama, câteodată, din valorile mici de o regulă pe care o urmează numerele, sau măcar putem determina mai ușor configurații pentru care jocul are sau nu strategie de câștig, fapt care ne poate ajuta în descoperirea rezolvării generale.

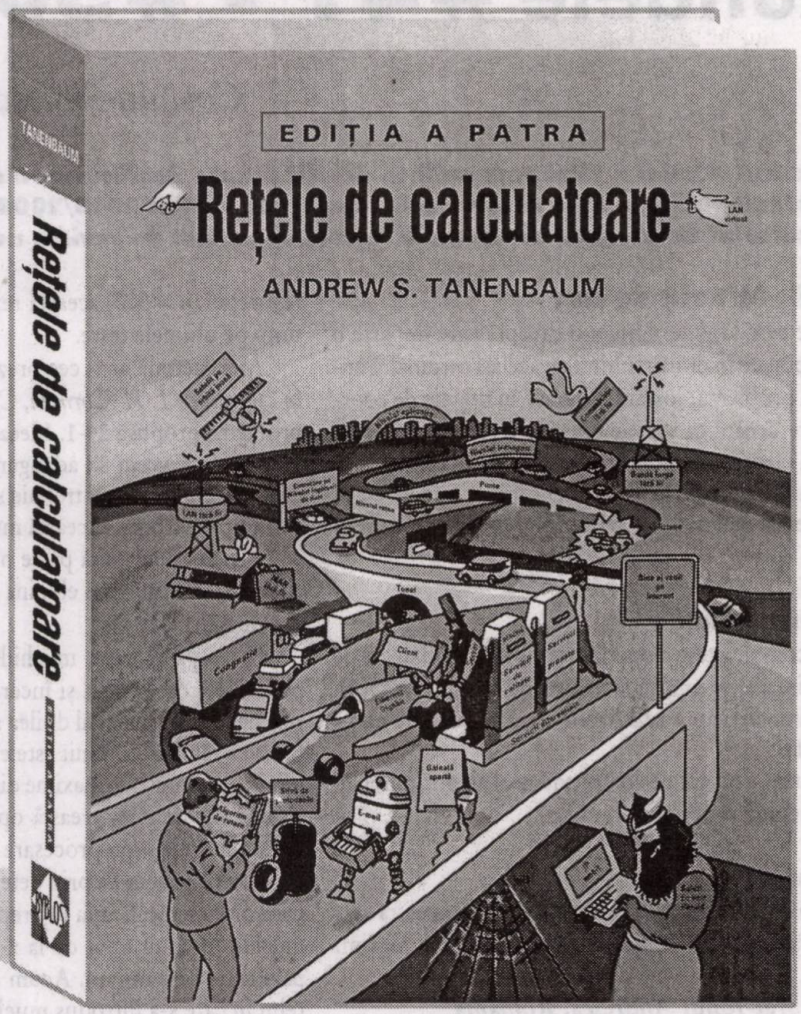
Bibliografie

1. ***, *colecția GInfo*
2. Mihai Oltean, *Programarea Jocurilor Matematice*, Editura Albastră, Cluj-Napoca, 1996
3. Thomas S. Ferguson, *Game Theory Text* (*online*) **http://www.math.ucla.edu/~tom/gamescourse.html**
4. **http://www.ams.org/new-in-math/cover/games1.html**
5. **http://www.cut-the-knot.com**
6. **http://www.mathworld.wolfram.com**
7. **http://ipsc.ksp.sk/**

Cosmin-Silvestru Negruşeri este student în anul III la Universitatea Babeş-Bolyai din Cluj-Napoca. El poate fi contactat prin e-mail la adresa **kosmin_2000@yahoo.com**.

Cine are carte, are parte!

... de înțelegerea rețelelor!



Cea mai bună introducere din lume în rețele de calculatoare
- complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o reducere semnificativă!

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65; 0721-927.499

Rundele #41 - #50

Cosmin-Silvestru Negruşeri

În cadrul acestui articol vom publica detalii despre desfăşurarea celor 10 probleme propuse spre rezolvare la semifinala ediţiei 2003/2004 a concursului de programare Bursele Agora, organizat de revista noastră.

P040429: Minele de aur

Problema ne cere să determinăm o dreaptă care împarte o mulţime de puncte în două mulţimi cu acelaşi cardinal. Pentru a rezolva problema, sortăm punctele în funcţie de coordonata x şi returnăm ca soluţie dreapta verticală care trece prin punctul cu indexul n din şirul sortat al punctelor. Este evident că această dreaptă conţine cel putin n puncte pe ea şi în stânga ei, şi mai conţine cel putin n puncte pe ea şi în dreapta ei.

Analiza complexităţii

Ordinul de complexitate a operaţiilor de citire a datelor de intrare şi găsire a elementului median dintr-un şir este $O(n)$.

Operaţia de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n)$.

Au rezolvat corect...

Dintre cei 25 de concurenţi care au participat la această etapă a concursului, şase au reuşit să obţină punctajul maxim.

Mugurel-Ionuţ Andreica, România
Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Marius Nicolae, România
Claudiu Rad, România

P040430: Rivali

Aceasta problemă cere determinarea diferenţei costurilor dintre arborele de cost minim şi al doilea arbore de cost minim al unui graf.

O primă metodă ar fi determinarea arborelui de cost minim şi eliminarea câte unei muchii, care aparţine acestui arbore, din graf, şi apoi determinarea noului arbore parţial pentru noul graf. Acum putem determina diferenţa dintre costul primului arbore obţinut şi costul celui mai mic obţinut pentru cei n-1 arbori determinaţi ulterior. Datorită restricţiei (n = 500) această rezolvare nu s-ar fi încadrat în timp pe ultimele teste.

Altă metodă ar fi cea prezentată în cartea *Introducere în algoritmi, T. H. Cormen, C. E. Leiserson, R. R. Rivest* în problema propusă 24-1. Ideea prezentată acolo este următoarea: dacă vrem să adăugam muchia (i, j) arborelui de cost minim, mai întâi trebuie să eliminăm o muchie pentru a obţine un arbore. Această muchie nou introdusă ar induce un ciclu, pe de altă parte noul nostru arbore trebuie să aibă cost minim, vom elimina muchia maximă din ciclul indus de muchia (i, j).

Parcurgând toate muchiile care nu aparţin arborelui parţial de cost minim şi încercând să le adăugăm la arborele nostru, obţinem al doilea arbore parţial de cost minim. Ceea ce rămâne de făcut este eficientizarea operaţiei de determinare a muchiei maxime din ciclul indus de muchia (i, j). Putem să facem ca această operaţie să se execute în timp constant după o preprocesare a grafului.

Realizăm aceasta prin determinartea valorilor unei matrice M cu semnificaţia $M_{i,j}$ reprezintă costul maxim al unei muchii de pe drumul de la nodul i la nodul j în arborele parţial de cost minim. Acum determinarea costului arborelui în care s-a introdus muchia (i, j) este:

$$cost\_arbore\_iniţial + cost(i, j) - M_{i,j}.$$

Determinarea câte unei linii din matricea M se poate face prin efectuarea unei parcurgeri în adâncime pe arborele parţial de cost minim, pornind de la nodul ale cărui linii din matrice vrem să o determinăm.

Analiza complexităţii

Ordinul de complexitate al operaţiei de citire a datelor de intrare este $O(m)$.

Determinarea arborelui parţial de cost minim cu ajutorul algoritmului lui *Prim* are ordinul de complexitate $O(n^2)$. Preprocesarea unei linii din matricea M are complexitatea unei parcurgeri în lăţime a unui graf cu n-1 muchii care este egală cu $O(n)$, deci ordinul de complexitate al preprocesării este $O(n^2)$.

Ordinul de complexitate al determinării celui de-al doilea arbore de cost minim este $O(m)$, după faza de preprocesare.

Ordinul de complexitate al operației de scriere a rezultatelor este $O(1)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n^2)$.

Au rezolvat corect...

Dintre cei 24 de concurenți care au participat la această etapă a concursului, șapte au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Adrian Diaconu, România
Haryanto Lego, Statele Unite ale Americii
Marius Nicolae, România
Mircea Pașoi, România
Mihail-Cosmin Pițt-Rada, România
Alexandru Popa, România

P040431: Cerc

În continuare vom prezenta trei metode de rezolvare pentru această problemă. Luăm două puncte consecutive de pe înfășurătoarea convexă a punctelor date (pentru simplitate putem alege cel mai de jos punct și punctul cel mai din dreapta față de cel mai de jos punct), fie ele A și B.

Considerăm unghiurile formate de celelalte n - 2 puncte cu punctele A și B. Dacă avem alte două puncte P și Q și relația $\angle APB \leq \angle AQB$, atunci punctul Q este situat în interiorul cercului circumscris triunghiului APB. Tragem concluzia că pentru a avea k puncte incluse într-un cerc, putem sorta descrescător cele n - 2 puncte în funcție de unghiul format cu punctele A și B, și putem returna cercul circumscris punctelor A, B și punctului cu indexul k - 2 din șirul sortat al celorlalte n - 2 puncte. Determinarea elementului cu indexul k - 2 se poate face cu un ordin de complexitate medie $O(n)$ folosind un algoritm de selecție aleatoare.

Ne este accesibilă și altă rezolvare datorită mărginirii domeniului punctelor la coordonata maximă 10000. Ca și mai sus, luăm două puncte consecutive A și B de pe înfășurătoare și apoi facem o căutare binară pentru a determina dimensiunea razei cercului care conține k puncte în interior. are cele două puncte A și B pe circumferință, iar centrul este în semiplanul în care sunt celelalte n - 2 puncte.

O a treia rezolvare, ar fi să considerăm centrul cercului punctul de coordonate (0, 0) și apoi să calculăm distanțele de la fiecare punct la centrul cercului. Acum soluția care trebuie returnată este a k-a distanță din șirul sortat al distanțelor. Acest număr îl găsim folosind metoda de selecție aleatoare.

Primele două rezolvări determină un cerc care trece prin cel puțin trei puncte din cele n puncte date.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(n)$, iar operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În cazul primelor două metode de rezolvare prezentate, operația de determinare a punctelor A și B are ordinul de complexitate $O(n)$.

În cadrul primei metode de rezolvare, determinarea unghiurilor formate de cele n - 2 puncte rămase cu punctele A și B are ordinul de complexitate $O(n)$, iar determinarea elementului k - 2 din șirul sortat al unghiurilor are ordinul de complexitate $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme, folosind prima metodă prezentată, este $O(n)$.

Pentru cea de-a doua metodă de rezolvare, numărul de pași pentru determinarea razei r a cercului este $O(\log r)$, și la fiecare pas trebuie să numărăm câte puncte sunt interioare cercului de rază r care trece prin punctele A și B, pas care are ordinul de complexitate $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme, folosind cea de-a doua metodă prezentată, este $O(n \cdot \log r)$.

Determinarea distanțelor de la cele n puncte până la punctul de coordonate (0, 0) pentru cea de-a treia metodă de rezolvare, are ordinul de complexitatea $O(n)$, iar determinarea elementului al k-lea din șirul sortat al distanțelor are ordinul de complexitate $O(n)$. Deci, ordinul de complexitate al algoritmului de rezolvare folosind această metodă este $O(n)$.

A rezolvat corect...

Dintre cei 13 de concurenți care au participat la această etapă a concursului, doi au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Marius Dumitran, România

P040432: Parola

Această problemă este similară cu problema *Parola Ascunsă* propusă spre rezolvare la etapa regională a concursului *ICPC* organizat de *ACM* anul trecut. O rezolvare cu ordinul de complexitate $O(n \cdot \log^2 n)$ a fost prezentată în revista *GInfo* 13/8 de către *Mihai Stroe*.

Vom prezenta în continuare o soluție care are ordinul de complexitate $O(n \cdot \log n)$. Fie S_i șirul rotit care începe la indexul i.

Trebuie să determinăm rotația care este minimă în ordine lexicografică. Pentru a face aceasta, vom compara șirurile în forma în care am organiza confruntările la un turneu de tenis, câștigătorul turneului fiind rotația care este minimă în ordine lexicografică.

Să vedem ce se întâmplă într-un meci al turneului:

Fie șirul inițial împărțit în trei secvențe A, B și C. La pasul actual trebuie să comparăm șirul ABC cu șirul BCA, unde A și B au aceeași lungime.

Dacă $A < B$, atunci ABC este câștigătorul meciului dintre ABC și BCA.

Dacă $A > B$, atunci BCA este câștigătorul meciului dintre ABC și BCA.

Dacă $A = B$, atunci îl luăm pe ABC ca și câștigător. Dacă decizia noastră este greșită și $ABC > BCA$, atunci înseamnă că $B > C$ și $A > C$, deci la un moment ulterior al turneului, BCA ar fi înfrânt de CAB sau de învingătorul lui CAB, deci, prin ignorarea lui BCA nu am alterat rezultatul final.

Când comparăm șirul S_i cu șirul S_j trebuie să comparăm $j - i$ elemente (practic subsecvențele notate cu A și B).

La runda k a turneului șirurile care trebuie comparate vor fi la distanța cel mult 2^{k-1} și vor fi $n/2^k$ șiruri de comparat.

Deci complexitatea unei runde este $O(n)$ și numărul total de runde este $\log n$.

Algoritmul în pseudocod al rezolvării este următorul:

```
Pentru i =1, n execută:
   //în w păstrăm învingătorii rundei anterioare
   adaugă pe i la w
sfârșit pentru
cât timp |w| > 1 execută:
   inițializează(t)
   cât timp |w| > 1 execută:
      compară primele w2 - w1 litere ale rotației w1 și
      rotației w2
      dacă (w1 > w2) atunci
         adaugă pe w2 la t
      altfel
         adaugă pe w1 la t
   sfârșit dacă
         șterge din w pe w1 și w2
  sfârșit cât timp
   dacă |w| = 1 atunci
      adaugă pe w1 la t
  sfârșit dacă
   w ← t
sfârșit cât timp
```

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(n)$.

Ordinul de complexitate al algoritmului de determinare a șirului cerut este $O(n \cdot \log n)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n \cdot \log n)$.

Au rezolvat corect...

Dintre cei 25 de concurenți care au participat la această etapă a concursului, patru au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Marius Dumitran, România
Mircea Pașoi, România

P040433: Coridorul

Un cerc de rază R dată nu poate trece dintr-un capăt al coridorului în altul, dacă există o succesiune de puncte în care fiecare două puncte sunt la distanță mai mică decât $2 \cdot R$, primul punct este la distanță mai mică decât $2 \cdot R$ de marginea de *Nord* a coridorului, iar ultimul punct este la distanță mai mică de $2 \cdot R$ de marginea de *Sud* a coridorului.

Determinarea existenței unui astfel de lanț pentru o rază dată se face ușor printr-o căutare în lățime pe graful în care se consideră existența unei muchii între două puncte atunci când distanța dintre ele este mai mică decât $2 \cdot R$. De aici se conturează ideea de a face o căutare binară pentru a afla dimensiunea razei maxime. Dacă nu avem grijă la implementare, o astfel de rezolvare poate depăși ușor timpul de execuție impus în enunț.

O altă rezolvare care se bazează pe aceeași idee este determinarea pentru fiecare punct a numărului d_i cu semnificația că d_i reprezintă valoarea minimă a muchiei maxime de pe orice lanț cu care se poate ajunge de la peretele din *Nord* la punctul i. Acest tablou poate fi determinat ușor, folosind un algoritm similar algoritmului *Dijkstra*.

Rezultatul nostru este egal cu:
min(max(d_i, distanța de la punctul i la peretele de sud))/2.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(k)$.

Numărul de iterații a căutării binare pentru prima metodă de rezolvare este $O(\log R)$, iar ordinul de complexitate al căutării în adâncime este $O(k^2)$. Deci acest pas din rezolvare are ordinul de complexitate $O(k^2 \cdot \log R)$.

În cazul celei de-a doua metode de rezolvare ordinul de complexitate al algoritmului este același cu cel al algoritmului lui *Dijkstra* de găsire a drumurilor minime, adică $O(k^2)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(k^2 \cdot \log R)$, dacă folosim prima metodă și $O(k^2)$, dacă folosim cea de-a doua metodă.

A rezolvat corect...

Dintre cei 14 de concurenți care au participat la această etapă a concursului, unul singur a reușit să obțină punctajul maxim.

Silviu-Ionuț Gănceanu, România

P040434: Ordinea

Problema este o variantă a problemei clasice numită problema lui *Josephus*. Dimensiunea lui N a fost aleasă destul de mică și o rezolvare care folosea o listă circulară înlănțuită și scotea pe rând din listă câte un copil ar fi luat 86 de puncte, ceea ce este destul de mult având în vedere că pe celelalte probleme propuse se scoteau mai dificil puncte fără rezolvări optime.

Să prezentăm acum două soluții bazate pe același principiu care ar fi luat punctaj maxim.

Ideea este că la fiecare pas se scoate un element din listă și trebuie găsit un element cu un index determinat (dacă elementele eliminate la pasul curent k era la poziția i, atunci elementul de eliminat la pasul următor este pe poziția $(i - 1 + k)$ modulo $(n - i) + 1$). Dacă am folosi un șir de indecși, atunci găsirea unui element de index p cere timp constant, iar ștergerea unui element de pe poziția p ar avea complexitate medie $O(n)$, complexitatea operațiilor este prea mare. Prezentăm în continuare două structuri de date care pot fi folosite cu succes în rezolvarea problemei.

Prima structură de date: Un șir num_i cu semnificația numărul elevilor cu numărul de ordine între i și $i + [\sqrt{n}]$ care mai sunt în listă. Pentru a elimina un element de index p din listă trebuie să actualizăm valorile lui *num* pentru indecșii $p - [\sqrt{n}]$, ..., p. Pentru a găsi elementul al p-lea care nu a fost încă eliminat, parcurgem în salturi de $[\sqrt{n}]$ și ajungem la un index i, astfel încât al p-lea element se află între i și $i + \sqrt{n}$, deci pentru găsirea celui de-al p-lea element care nu a fost încă eliminat avem maxim $2 \cdot \sqrt{n}$ operații.

A doua structură de date: Folosim un arbore binar complet care conține 2^x noduri astfel ca 2^x să fie mai mare sau egal cu $2 \cdot n$. Nodul rădăcină păstrează informație despre elementele 1, ..., n ale șirului, fiul stâng păstrează informații despre elementele 1, ..., $[(n + 1)/2]$ și fiul drept despre elementele $[(n + 1)/2] + 1$, ..., n, deci un nod fiu stâng păstrează jumătatea din stângă a numerelor păstrate de nodul tată, iar fiul drept păstrează jumătatea dreaptă a numerelor păstrate de nodul tată Informația păstrată pentru nodul asociat intervalului (i, j) este numărul de elemente, cu indecși între i și j inclusiv, existente în lista noastră. Ștergerea din structura noastră de date are complexitatea $O(\log n)$ pentru că un copil este cuprins în $\log n$ intervale $[i, j]$ monitorizate în arbore. Găsirea elementului de index p din listă se face similar cu procedeul de determinare al elementului p dintr-un șir de la selecția aleatoare, complexitatea acestei operații fiind și ea $O(\log n)$.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(1)$, iar operația de scriere a rezultatelor are ordinul de complexitate $O(n)$.

Căutarea și scoaterea celor n elemente din structura de date folosind prima structură de date prezentată are ordinul de complexitate $O(n \cdot \sqrt{n})$.

În cazul celei de-a doua structuri de date căutarea și scoaterea celor n elemente din structura de date $O(n \cdot \log n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n \cdot \sqrt{n})$ dacă folosim prima structură de date și $O(n \cdot \log n)$, dacă folosim cea de-a doua structură.

Au rezolvat corect...

Dintre cei 32 de concurenți care au participat la această etapă a concursului, șapte au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Adrian Cârcu, România
Liviu Ciortea, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Mircea Pașoi, România
Sorin Stancu-Mara, România

P040435: Pioni

Această problemă este o aplicație de bază pentru numerele *Sprague-Grundy*, pentru o analiză a ei citiți articolul *Teoria jocurilor: numerele SPRAGUE GRUNDY*, de la pagina 19.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(m + t \cdot n)$.

Ordinul de complexitate al sortării topologice a nodurilor grafului este $O(n + m)$. Determinarea valorilor *Sprague Grundy* pentru noduri are complexitatea $O(n \cdot$ valoarea maximă $SG) \leq O(n^2)$.

Ordinul de complexitate al rezolvării unui caz este $O(n)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(t)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este mai mic decât $O(m + n \cdot t + n^2 + t)$.

Au rezolvat corect...

Dintre cei 10 de concurenți care au participat la această etapă a concursului, șase au reușit să obțină punctajul maxim.

Chattopadhyay Amit, India
Mugurel-Ionuț Andreica, România
Marius Dumitran, România
Silviu-Ionuț Gănceanu, România
Mircea Pașoi, România
Adrian Vladu, România

P040436: Matrice binară

Orice două dreptunghiuri cu laturile paralele cu axele de coordonate, care nu se intersectează pot fi despărțite printr-o dreaptă verticală sau una orizontală. Să ne ocupăm acum de determinarea ariei maxime care poate fi acoperită de două dreptunghiuri care conțin numai valoarea 0, dreptunghiuri care pot fi despărțite printr-o dreaptă orizontală (cazul cu dreapta verticală se rezolvă analog).

Dacă am avea pentru fiecare linie i calculate numerele u_i (aria maximă a unui dreptunghi care conține numai 0, situat mai sus de linia $i + 1$) și l_i (aria maximă a unui dreptunghi care conține numai 0, situat mai jos de linia i), atunci rezultatul nostru ar fi $\max(u_i + l_i, \forall 1 \leq i \leq n)$. Să vedem acum, cum putem calcula u_i, calcularea lui l_i făcându-se analog.

Am redus astfel problema la determinarea ariei maxime a unui dreptunghi care conține numai 0. Această pro-

blemă se poate rezolva în mai multe moduri, rezolvarea optimă având complexitatea $O(n \cdot m)$, unde m și n sunt dimensiunile matricei în care căutăm dreptunghiul. Ea a fost prezentată pe lista de discuții a lui Cătălin Frâncu, fost olimpic, și poate fi găsită la adresa:
http://probleme.francu.com/arhiva/R0038/rezolv.html.

O rezolvare care s-ar fi încadrat în timp ar fi fost următoarea: pentru elementul j al liniei i avem calculat numărul h_j care reprezintă înălțimea bandei continue de zerouri care pornește de la celula (i, j) și merge în sus. Aria dreptunghiului care conține numai 0 și care are colțul din dreapta-jos în celula (i, j) și colțul din stânga-jos în (i, k) este $(j - k + 1) \cdot \min(h_p)$, $k \le p \le j$. Dacă avem șirul h calculat pentru linia curentă, putem determina aria maximă a unui dreptunghi care conține numai 0, cu colțul de dreapta-jos în (i, j) cu un algoritm care are ordinul de complexitate $O(n)$.

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare este $O(m \cdot n)$.

Ordinul de complexitate al determinării ariei maxime a unui dreptunghi situat mai sus de linia $i + 1$ este $O(n) \cdot O(n \cdot i)$. Complexitatea totală a determinării elementelor șirului u este $O(n^3)$.

Operația de scriere a rezultatelor are ordinul de complexitate $O(1)$.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n^3)$. Această complexitate poate fi redusă la $O(n^2)$, dacă rezolvăm subproblema determinării ariei maxime a unui dreptunghi care conține numai 0 urmând algoritmul prezentat pe "*Lista lu' Frâncu*".

Au rezolvat corect...

Dintre cei 9 concurenți care au participat la această etapă a concursului, patru au reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România
Silviu-Ionuț Gănceanu, România
Marius Nicolae, România
Sorin Stancu-Mara, România

P040437: Numere

În această problemă accentul se pune mai mult pe atenție decât pe partea de algoritmică. Trebuie să tratăm multe cazuri. O funcție care ne va fi utilă în rezolvarea problemei este cea a determinării la ce poziție în șir am scris numărul k, această funcție va avea complexitatea $O(\log N)$, unde N = 1.000.000.000.

Vom da acum câteva cazuri:

- `7989910010110` – `99` generează secvența căutată;
- `2345612345` – numărul generator este `123456`;
- `1771` – numărul generator este `717`;
- `567234` – numarul generator este `234567`.

Ultimele trei cazuri sunt mai dificile decât primul.

La fiecare caz determinăm poziția în șir a numărului care generează secvența folosind funcția ajutătoare (numărul total de cazuri este mic, deci rezolvarea pentru dimensiunea testelor va avea timp de rulare foarte mic).

Analiza complexității

Ordinul de complexitate al operației de citire a datelor de intrare și scriere a rezultatelor este $O(t)$.

Rezolvarea are complexitatea $O(nr \cdot \log N \cdot t)$, unde nr reprezintă numărul de cazuri care trebuie tratate, iar t reprezintă numărul maxim de teste.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(nr \cdot \log N \cdot t)$.

A rezolvat corect...

Dintre cei 15 de concurenți care au participat la această etapă a concursului, unul singur a reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România

P040438: Cercuri

O rezolvare polinomială pare să fie greu de găsit pentru această problemă, și în cazul a numai două cercuri apar o grămadă de cazuri care trebuie tratate. Rezolvarea unei variante mai simple a problemei ne va ajuta în soluția problemei generale. Varianta mai simplă care este ușor de rezolvat este determinarea pătratului de latură minimă cu laturile paralele cu axele de coordonate care conține în el cercurile date. Acum, pentru a determina un pătrat de latură minimă care acoperă cercurile, rotim puțin câte puțin sistemul de coordonate față de origine, și pentru fiecare rotație determinăm latura minimă a pătratului care conține cercurile și are laturile paralele cu axele de coordonate. Dacă unghiul de rotație este destul de mic vom obține latura minimă a pătratului cu o precizie de 8 zecimale.

Analiza complexității

Ordinul de complexitate a operațiilor de citire a datelor de intrare și scriere a rezultatelor este $O(1)$.

Determinarea laturii pătratului cerut are ordinul de complexitate $O(\pi/2/r)$, unde r reprezintă numărul de rotiri ale sistemului de coordonate.

Trebuie să fim atenți ca numărul de rotații să fie suficient de mic pentru ca rezolvarea noastră să se încadreze în timp, și suficient de mare ca să obținem lungimea laturii cu o precizie de 8 zecimale.

În concluzie ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(\pi/2/r)$.

A rezolvat corect...

Dintre cei 15 de concurenți care au participat la această etapă a concursului, unul singur a reușit să obțină punctajul maxim.

Mugurel-Ionuț Andreica, România

Bursele Agora 2003/2004 Top 50

1. Mugurel-Ionuț Andreica, București	4424
2. Adrian Cârcu, Bistrița	3718
3. Mircea Pașoi, Ploiești	3469
4. Silviu-Ionuț Gănceanu, Bârlad	3220
5. Marius Dumitran, București	3155
6. Liviu Ciortea, București	3148
7. Haryanto Lego, Statele Unite ale Americii	3034
8. Kiril Minkov, Bulgaria	2977
9. Mihail-Cosmin Pițt-Rada, Dr. Tr. Severin	2818
10. Chattopadhyay Amit, India	2758

11. Adrian Vladu, București	2669	31. Valentin Stanciu, București	822
12. Adrian Diaconu, București	2523	32. Victor Podeanu, București	750
13. Vasanth Jai, India	2411	33. Andrei-Ionuț Tudor, Slobozia	712
14. Csaba Pătcaș, Oradea	2387	34. Eduard-G. Băzăvan, Dr. Tr. Severin	709
15. Claudiu Rad, Deva	2351	35. Stelian Kelemen, Bistrița	688
16. Sorin Stancu-Mara, București	2090	36. Ioan-Dan Docleanu, Nădlac	686
17. Alexandru Peptan, Dr. Tr. Severin	2005	37. Dragoș Cavulea, Botoșani	634
18. Kazmenko Ivan, Federația Rusă	1966	38. Adina Pampu, Făgăraș	621
19. Alexandru Popa, București	1882	39. Dalian-Valeriu Moraru, București	617
20. Sergiu Hlihor, Lugoj	1635	40. Irinel Besliu, Vaslui	592
21. Tudor-Alexandru Rabaea, Constanța	1599	41. Sorin Vultureanu	570
22. Marius Nicolae, Slatina	1465	42. Adrian-Romulus Ciripan, Alba-Iulia	540
23. Mihai-Vlad Pantiș, Cluj-Napoca	1267	43. Călin Bindea, Bistrița	497
24. Mihai Moalfă, Sîngeorz-Băi	1221	44. Vlad Petcu, Boldești-Scăieni	494
25. Cristian Cernat, Târgu Jiu	1203	45. Renald , Indonezia	460
26. Andrei Gönczi, Arad	1192	46. Cosmin Molea, Bascov	450
27. Andrei Janca, Dr. Tr. Severin	1097	47. Bogdan-Gabriel Ionescu, Craiova	404
28. Ciprian Păduraru, Bârlad	911	48. Johannes Slotta, Germania	396
29. Iulian Blajuț, Luncani	869	49. Ștefan Gheorghe, București	383
30. Claudiu-Dan Gheorghe, Slobozia	841	50. Cosmin Matei, Ploiești	380

Bursele Agora 2003/2004!

BILANȚ

Cu excepția etapei finale, putem spune că ediția din acest an a concursului de programare Bursele Agora organizat de revista noastră s-a încheiat. În continuare vă vom prezenta evoluția concurenților de-a lungul celor 50 de runde.

Concurenții

În cadrul celei de-a cincia ediții a concursului de programare *Bursele Agora* au fost propuse spre rezolvare, de-a lungul celor 50 de runde, 50 de probleme cu diferite grade de dificultate.

La ediția din acest an a concursului de programare *Bursele Agora* au participat 395 de concurenți, dintre care 350 sunt din *România*.

Concurenții au reușit să facă față problemelor obținând, în mare parte, punctaje satisfăcătoare.

Dintre cei 395 de înscriși la ediția din acest an a concursului de programare, doar aproape jumătate au trimis soluții la cel puțin o problemă propusă spre rezolvare și 90 dintre concurenți au reușit să obțină mai mult de 100 de puncte din cele 5000 posibile.

Fiecare dintre cele 50 de probleme propuse, 48 dintre ele au fost rezolvate corect de cel puțin unul dintre concurenți.

Dintre cei aproape 400 de concurenți, 76 dintre ei au rezolvat corect cel puțin o problemă.

Cum era de așteptat, nu au ezitat să apară defecțiuni tehnice și greșeli, atât din partea noastră, cât și din partea concurenților.

Doi concurenți aflați în primele douăzeci de locuri au fost descalificați în urma încălcării regulamentului de participare prin trimiterea de soluții identice la câteva runde ale semifinalei concursului.

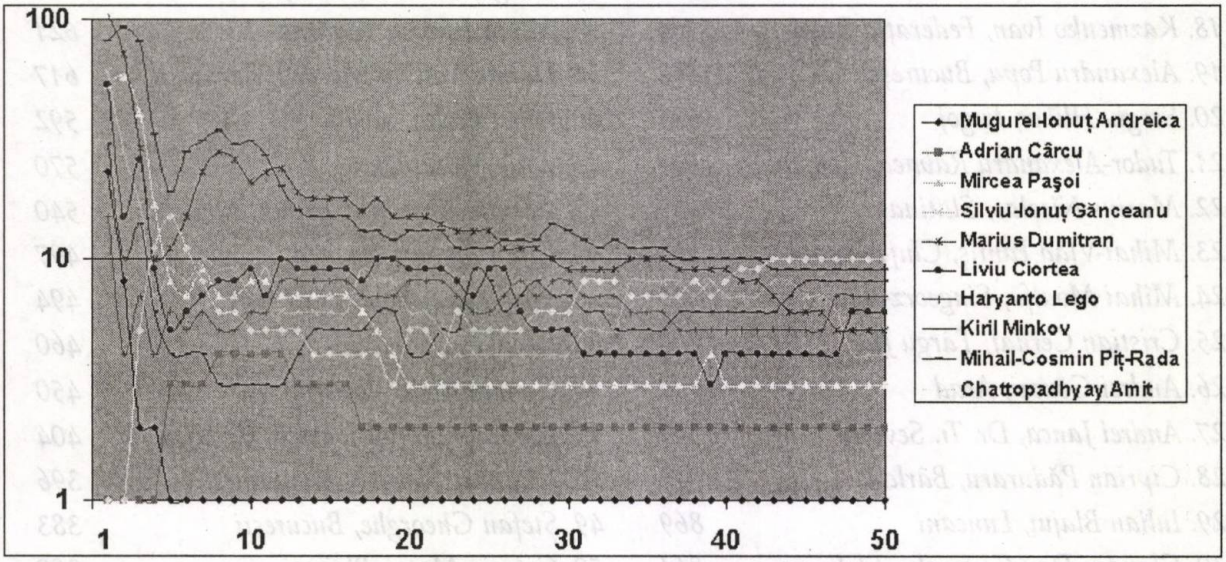
Performanțe

Doar doi concurenți au reușit să rezolve corect mai mult de jumătate dintre problemele propuse. Aceștia sunt:

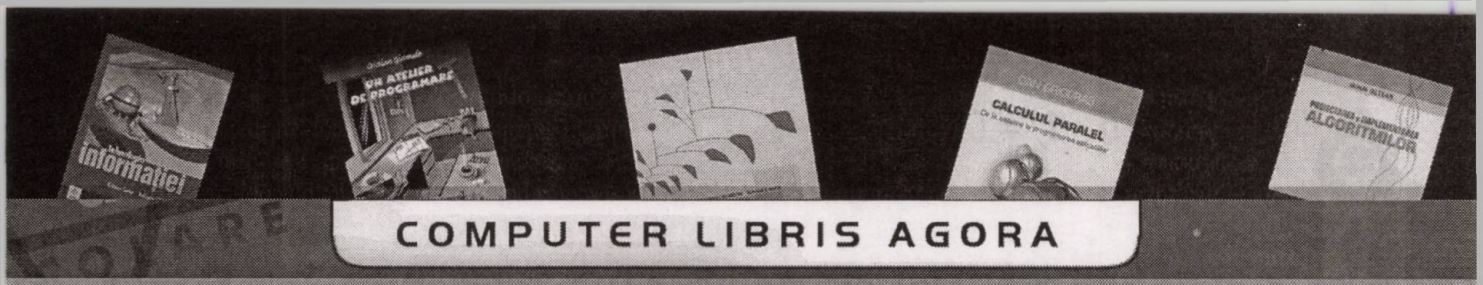
Mugurel-Ionuț Andreica	*41 de probleme*
Adrian Cârcu	*31 de probleme*

În graficul de mai jos este prezentată evoluția din timpul concursului a primilor 10 clasați la sfârșitul celor 50 de runde.

Se poate observa foarte ușor că *Mugurel-Ionuț Andreica* se află în fruntea clasamentului începând cu runda 5.



Evoluția celor mai buni participanți în timpul concursului

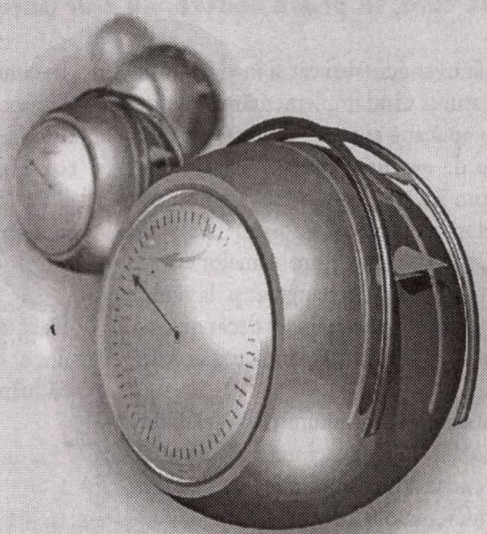


COMPUTER LIBRIS AGORA

DAN GRIGORAȘ

CALCULUL PARALEL

De la sisteme la programarea aplicațiilor



Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate,dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performanță și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 0264 - 592422

Optimizarea programelor folosind OPERAȚII PE BIȚI

Cosmin-Silvestru Negruşeri

În cadrul acestui articol vă vom prezenta câteva metode prin care programele se pot optimiza dacă utilizăm eficient operațiile pe biți sau folosim operații pe biți în locuri în care, la prima vedere, nu s-ar părea că ar fi necesare.

De cele mai multe ori, atât în concursuri cât și în viața de zi cu zi a programatorului, atunci când implementăm o metodă care este folosită de o aplicație și vrem să facem această metodă eficientă ca timp de execuție, suntem învățați din școală (sau ar trebui să fim învățați, chiar dacă unii dintre profesori consideră că cel mai important algoritm învățat în liceu este *backtraking*-ul, soluția tuturor problemelor) să ne uităm la complexitatea algoritmului implementat și dacă observăm că în practică algoritmul este mai încet decât ne dorim noi să fie să încercăm să găsim un algoritm cu un ordin de complexitate mai mic.

Nu trebuie să uităm că ceea ce numim complexitatea unui algoritm este o aproximare a vitezei unui algoritm și nu o măsură absolută. Pentru dimensiuni mici ale datelor, câteodată nu se observă diferența dintre $O(n)$ și $O(n \cdot \log n)$ sau $O(n^{3/2})$. Mie personal mi s-a întâmplat ca la implementarea soluției unei probleme care în engleză se numește "*Bottleneck Minimal Spanning Tree*" să observ că rezolvarea în $O(m \cdot \log m)$ (folosind algoritmul de găsire a arborelui parțial de cost minim al lui *Kruskal*) să fie mai rapidă decât rezolvarea mai laborioasă în $O(m)$ a acestei probleme. Deci trebuie dată o atenție egală cu cea acordată complexității algoritmului și dimensiunii factorilor constanți care apar.

În continuare vom încerca să găsim soluții mai rapide decât cele naive pentru unele operații de bază (toate operațiile vor fi implementate pentru întregi pozitivi reprezentați pe 32 de biți).

Problema 1

Să se determine numărul de biți 1 din reprezentarea binară a numărului n.

Rezolvarea naivă a acestei probleme ar consta în parcurgerea secvențială a biților lui n.

În continuare vă prezint această rezolvare:

```
int count(long n){
  int num = 0;
  for (int i = 0; i < 32; i++)
    if (n & (1 << i)) num++;
  return num;
}
```

Dacă ne uităm cu atenție și analizăm rezultatul operației `n & (n - 1)` putem obține o soluție mai bună.

Să luăm un exemplu:

```
n         = (11011101010000)2
n - 1     = (11011101001111)2
n & (n-1) = (11011101000000)2
```

Se vede clar de aici că efectul operației `n & (n - 1)` este anularea celui mai nesemnificativ bit cu valoarea 1.

De aici ideea algoritmului:

```
int count(long n){
  int num = 0;
  if (n)
    while (n &= n - 1) num++;
  return num;
}
```

Dar este acest algoritm mai rapid? Am testat pentru toate numerele de la 1 la 2^{24} și rezultatele au fost 2,654 secunde folosind metoda naivă și 0.821 folosind a doua metodă.

Rezultatul mult mai bun al celei de-a doua metode se bazează în principal pe faptul că ea execută un număr de pași egali cu numărul de biți cu valoarea 1 din număr, deci în medie jumătate din numărul de pași efectuați de prima metodă.

Problema 2

Să se determine paritatea numărului de biți de 1 din reprezentarea binară a unui număr *n*.

Din cele prezentate mai sus se pot determina două metode evidente:

```
int parity(long n){
  int num = 0;
  for (int i = 0; i < 32; i++)
    if (n & (1 << i)) num ^= 1;
  return num;
}
```

```
int parity(long n){
  int num = 0;
  if (n)
    while (n &= n - 1) num ^= 1;
  return num;
}
```

Putem obține o a treia rezolvare făcând câteva observații pe un exemplu:

Considerăm n = $(11011011)_2$. Rezultatul căutat este dat de valoarea 1 ^ 1 ^ 0 ^ 1 ^ 1 ^ 0 ^ 1 ^ 1.

Împărțim pe *n* în partea lui superioară și partea lui inferioară:

```
1 1 0 1
  ^
1 0 1 1
  =
0 1 1 0
```

Aplicăm asupra rezultatului același procedeu:

```
0 1
 ^
1 0
 =
0 1
```

și

```
1
^
1
=
0
```

Să scriem algoritmul care reiese din acest exemplu, luând în considerare faptul că numărul *n* este reprezentat pe 32 de biți:

```
int parity(long n){
  n = ((0xFFFF0000 & n) >> 16) ^ (n & 0xFFFF);
  n = ((0xFF00 & n) >> 8) ^ (n & 0xFF);
  n = ((0xF0 & n) >> 4) ^ (n & 0xF);
  n = ((12 & n) >> 2) ^ (n & 3);
  n = ((2 & n) >> 1) ^ (n & 1);
  return n;
}
```

Deși constanta multiplicativă este mai mare, numărul de operații are ordin logaritmic față de numărul de biți ai unui cuvânt al calculatorului.

Problema 3

Să se determine cel mai puțin semnificativ bit de 1 din reprezentarea binară a lui *n*.

Rezolvarea naivă se comportă bine în medie, deoarece numărul de cicluri până la găsirea unui bit cu valoarea 1 este de obicei mic, dar din cele discutate mai sus putem găsi ceva mai bun.

Așa cum am arătat n & (n - 1) are ca rezultat numărul *n* din care s-a scăzut cel mai puțin semnificativ bit.

Folosind această idee obținem următoarea funcție:

```
int low1(long n){
  return n ^ (n & (n - 1));
}
```

Exemplu:

n = $(11011000)_2$
n - 1 = $(11010111)_2$
n & (n - 1) = $(11010000)_2$
n = $(11011000)_2$
n ^ (n & (n - 1)) = $(00001000)_2$

Această funcție este foarte importantă pentru structura de date *Arbori Indexați Binar* prezentată într-un un articol mai vechi din *GInfo*.

Problema 4

Să se determine cel mai semnificativ bit cu valoarea 1 din reprezentarea binară a lui *n*.

Putem aplica și aici ideile prezentate mai sus: cea naivă și cea cu eliminarea biților, dar putem găsi și ceva mai bun.

O abordare ar fi cea a căutării binare (aplicabilă și în problema anterioară).

Verificăm dacă partea superioară a lui *n* este 0. Dacă nu este 0, atunci căutăm bitul cel mai semnificativ din ea, iar dacă este, ne ocupăm de partea inferioară, deci reducem la fiecare pas problema la jumătate.

```
int high1(long n) {
  long num = 0;
  if (!n) return -1;
  if (0xFFFF0000 & n) {
    n = (0xFFFF0000 & n)>>16;
    num += 16;
  }
  if (0xFF00 & n) {
    n = (0xFF00 & n) >> 8;
    num += 8;
  }
  if (0xF0 & n) {
    n = (0xF0 & n) >> 4;
    num += 4;
```

```
    }
    if (12 & n) {
      n = (12 & n) >> 2;
      num += 2;
    }
    if (2 & n) {
      n = (2 & n) >> 1;
      num += 1;
    }
    return 1 << num;
}
```

Problema 5

Să se determine indexul celui mai semnificativ bit de 1 din reprezentarea binară a lui *n*.

Metodele prezentate la rezolvarea problemei 4 pot fi folosite și aici, dar vom prezenta o nouă rezolvare.

Să luăm următorul șir de operații pentru un exemplu:

$n = (10000000)_2$
$n = n \mid (n >> 1)$
$n = (11000000)_2$
$n = n \mid (n >> 2)$
$n = (11110000)_2$
$n = n \mid (n >> 4)$
$n = (11111111)_2$

Se observă că aplicând o secvență asemănătoare de instrucțiuni cu cea de mai sus putem face ca un număr *n* să se transforme în alt număr care are un număr de biți de 1 egal cu 1 + indexul celui mai semnificativ bit cu valoarea 1 din *n*. De aici algoritmul este următorul:

```
int indexHigh1(long n){
  n = n | (n >> 1);
  n = n | (n >> 2);
  n = n | (n >> 4);
  n = n | (n >> 8);
  n = n | (n >> 16);
  return count(n) - 1;
}
```

Pentru ca această metodă să fie eficientă ne trebuie o metodă **count** eficientă.

Problema 6

Să se determine dacă *n* este o putere a lui 2 (întrebare interviu *Microsoft*).

```
int isTwoPower(long n){
  return (n & (n-1) == 0);
}
```

O altă abordare: preprocesarea

Operațiile prezentate mai sus sunt implementate în limbajele de asamblare, dar câteodată se folosesc algoritmi naivi și atunci o implementare inteligentă ajunge să fie mai rapidă decât instrucțiunea din limbajul de asamblare. Pentru unele supercalculatoare aceste instrucțiuni sunt instrucțiuni ale procesorului.

Se pot găsi alți algoritmi mai rapizi decât cei de mai sus, dar de obicei găsirea unui algoritm mai inteligent este mai grea decât folosirea unei abordări banale care aduce un câștig foarte mare: preprocesarea.

În continuare vom rezolva problemele prezentate anterior folosindu-ne de preprocesarea datelor.

Problema 1

Determinăm, pentru numerele de la 0 la 2^{16} - 1, un tablou `num`, în care `num[i]` este egal cu numărul de biți de 1 din reprezentarea binară a lui `i`.

De aici obținem soluția:

```
int count(long n){
  return num[n & 0xFFFF] + num[n >> 16];
}
```

Problema 2

Determinăm, pentru numerele de la 0 la 2^{16} - 1, un tablou `par`, în care `par[i]` este egal cu paritatea numărului de biți de 1 din reprezentarea binară a lui `i`.

De aici obținem soluția:

```
int parity(long n){
  return par[n & 0xFFFF] ^ par[n >> 16];
}
```

Problema 3

Determinăm, pentru numerele de la 0 la 2^{16} - 1, un tablou `l`, în care `l[i]` este egal cu cel mai nesemnificativ bit de 1 din reprezentarea binară a lui `i`.

De aici obținem soluția:

```
int low(long n){
  if (l[n & 0xFFFF]) return l[n & 0xFFFF];
  else return l[n >> 16] << 16;
}
```

Problema 4

Determinăm, pentru numerele de la 0 la 2^{16} - 1, un tablou `h`, în care `h[i]` este egal cu cel mai semnificativ bit de 1 din reprezentarea binară a lui `i`.

De aici obținem soluția:

```
int high(long n){
  if (h[n >> 16]) return h[n >> 16] << 16;
  else return h[n & 0xFFFF];
}
```

Problema 5

Determinăm, pentru numerele de la 0 la 2^{16} - 1, un tablou `iH`, în care `iH[i]` este egal cu indexul celui mai semnificativ bit de 1 din reprezentarea binară a lui `i` și este egal cu -1 pentru 0.

De aici obținem soluția:

```
int indexHigh(long n) {
  if (iH[n >> 16] != -1) return iH[n >> 16] + 16;
  else return iH[n & 0xFFFF];
}
```

Algoritmi mai serioși

Deși ceea ce am prezentat mai sus sunt mai degrabă trucuri de implementare decât algoritmi serioși și în general nu se acordă așa mare importanță detaliilor de genul acesta, câteodată asemenea trucuri se pot dovedi importante mai ales în timpul concursurilor.

Un exemplu ar fi problema rundei 12 de la concursul de programare *Bursele Agora*. Acolo se cerea găsirea numărului de dreptunghiuri conținute de o matrice binară *a* care au în colțuri 1.

Un algoritm de complexitate $O(n^2 \cdot m)$ ar fi fost următorul: se consideră fiecare pereche formată din două linii și se numără câte perechi $(a_{i,k}, a_{j,k})$ pentru care $a_{i,k} = 1$ și $a_{j,k} = 1$ există.

Fie acest număr x. Atunci numărul de dreptunghiuri care au colțurile pe aceste două linii va fi $x \cdot (x - 1) / 2$.

Un asemenea algoritm ar fi luat numai 64 de puncte deoarece restricțiile date erau mari ($n \leq 200$, $m \leq 2500$).

Putem rescrie condiția `a[i][k] == 1 && a[j][k] == 1` ca și `a[i][k] ^ a[j][k] == 1`).

Deci putem modifica rezolvarea anterioară în modul următor:

```
pentru fiecare i
  pentru fiecare j > i
    b = a[i] ^ a[j];
  sfârșit pentru
sfârșit pentru
  numaraBitiUnu(b)
```

Dacă în loc să păstrăm în `a[i][j]` un element al matricei binare, păstram 16 elemente, atunci în linia `b = a[i] ^ a[j]` se efectuează cel mult 2500 / 16 calcule în loc de 2500 de calcule. Acum ce a rămas de optimizat este metoda `numaraBitiUnu`.

Dacă folosim preprocesarea, atunci această metodă va fi compusă și ea 2500/16 calcule. Se observă că în acest fel am mărit viteza de execuție a algoritmului cu un factor de 16.

Alt exemplu ar fi problema rundei 17. În acea problemă se cere determinarea numărului de sume distincte care se pot obține folosind niște obiecte cu valori date folosind fiecare obiect pentru o sumă cel mult o dată.

Din nou o rezolvare directă, folosind marcarea pe un șir de valori logice n-ar fi obținut punctaj maxim. Pentru obținerea punctajului maxim următoarea idee ar fi putut fi folosită cu succes: șirul de valori logice se condensează într-un șir de întregi reprezentați pe 16 biți, și acum actualizarea se face asupra a 16 valori deodată. Deci și aici se câștigă un factor de viteză egal cu 8 (datorită detaliilor de implementare).

Un al treilea exemplu ar fi problema *Viteza* de la a 9-a rundă a concursului organizat de ***.campion*** anul acesta. Acolo se cerea determinarea celui mai mic divizor comun a două numere binare de cel mult 10000 de cifre.

După cum se știe, complexitatea algoritmului de determinare a celui mai mare divizor comun a două numere este logaritmică, acest fapt este adevărat atunci când operațiile efectuate sunt constante, dar în rezolvarea noastră apar numere mari, deci o estimare a numărului de calcule ar fi *maxn* · *maxn* · log *maxn* (împărțirea are în medie costul *maxn* · *maxn* dacă nu implementăm metode mai laborioase).

Acest număr este prea mare pentru timpul de rezolvare care ne este cerut. Pentru a evita împărțirea, care este operația cea mai costisitoare, putem folosi algoritmul de determinare a celui mai mare divizor comun binar care se folosește de următoarele proprietăți:

- *cmmdc*(*a*, *b*) = 2 * *cmmdc*(*a* / 2, *b* / 2), dacă *a* și *b* sunt numere pare;
- *cmmdc*(*a*, *b*) = *cmmdc*(*a*, *b* / 2), dacă *a* este impar;
- *cmmdc*(*a*, *b*) = *cmmdc*(*a* - *b*, *b*), dacă *a* este impar, *b* este impar și $a > b$.

Aplicând acest algoritm numărul de calcule s-a redus la *maxn* · log *maxn*.

Dar nici acest număr nu este suficient de mic, și deci pentru accelerarea algoritmului folosim ideea utilizată în cele două probleme de mai sus.Împachetăm numărul în întregi în pachete de câte 30 de biți, și atunci deplasarea la stânga (adică împărțirea la 2) și scăderea vor fi de 30 de ori mai rapide.

Să încercăm acum rezolvarea unei probleme în care eficiența este cea mai importantă, mai importantă decât lizibilitatea codului obținut. Problema are următorul enunț:

Se consideră un număr n. Să se determine numerele prime mai mici sau egale cu n, unde $n \leq 10.000.000$.

Practic această problemă ne va fi utilă în testarea rapidă a primalității numerelor mai mici sau egale cu *n* .

Putem încerca mai multe rezolvări, dar în practică cea mai rapidă se dovedește de obicei cea numită *ciurul lui Eratostene*. Ea se învață în școli încă din clasa a 6-a, deci nu o voi mai explica. Implementarea banală a algoritmului ar fi următoarea:

```
#define maxsize 10000000

char at[maxsize]; //vector de valori logice în care nume-
                  //rele prime vor fi marcate cu 0
int n;

int isPrime(int x){
  return (!at[x]) ;
}
```

```
void sieve(){
  int i, j;
  memset(at,0,sizeof(at)) ;
  for (i = 2; i <= maxsize; i++)
    if (!at[i])
      for (j = i + i; j <= maxsize; j += i)
        at[j]=1; //marcăm toți multipli lui i ca fiind
                 //numere prime
}
```

Observăm că jumătate din timpul folosit de noi la preprocesare este pierdut cu numerele pare. Marcându-le de la început vom putea ignora numerele pare în preprocesare:

```
#define maxsize 10000000

char at[maxsize];
int n;
int isPrime(int x){
  return (!at[x]) ;
}

void sieve(){
  int i, j;
  memset(at,0,sizeof(at));
  for (i = 4; i <= maxsize; i += 2)
    at[i] = 1;
  for (i = 3; i <= maxsize; i += 2)
    if (!at[i])
      for (j = i + i + i; j <= maxsize; j += i + i)
        at[j] = 1;
}
```

La marcarea multiplilor numărului prim *i* toate numerele până la $i \cdot i$ au fost deja marcate deoarece $i \cdot i$ este cel mai mic număr care nu este divizibil cu numere naturale mai mici sau egale cu $i - 1$.

Deci, avem o a treia versiune a programului:

```
#define maxsize 10000000
char at[maxsize];
int n;
int isPrime(int x){
  return (!at[x]) ;
}

void sieve(){
  int i, j;
  memset(at,0,sizeof(at));
  for (i = 4; i <= maxsize; i += 2)
    at[i] = 1;
  for (i = 3; i <= maxsize; i += 2)
    if (!at[i])
      for (j = i * i; j <= maxsize; j += i + i)
        at[j] = 1;
}
```

Ultima optimizare este optimizarea spațiului necesar. Într-un tip de date **char** putem împacheta opt valori logice și, punând un test suplimentar în metoda `isPrime` putem elimina și numerele pare, astfel vom avea nevoie de un spațiu de 16 ori mai mic.

```
#define maxsize 10000000

char at[maxsize/16];
int n;

int isPrime(int x) {
  if (!(x & 1))
    if (x==2) return 0 ;
    else return 1 ;
    else return (at[((x - 3) >> 1) >> 8] &
                 (1 << (((x - 3) >> 1) & 7))) ;
}
void sieve() {
  int i, j;
  memset(at, 0, sizeof(at)) ;
  for (i = 3; i <= maxsize; i += 2)
    if (!at[((i - 3) >> 1) >> 8] &
        (1 << (((i - 3) >> 1) & 7)))
      for (j=i * i ; j <= maxsize ; j += i + i)
        at[((i - 3) >> 1) >> 8] |=
          (1 << (((i - 3) >> 1) & 7))) ;
}
```

Concluzie

Asemenea optimizări ca și cele prezentate în cadrul acestui articol se pot dovedi foarte utile în unele cazuri, dar ele fac programul ilizibil.

Tocmai din acest motiv, asemenea trucuri trebuie aplicate numai în locurile critice ale codurilor sursă pentru a duce la o creștere semnificativă de viteză a aplicațiilor și trebuie documentate foarte bine, mai ales dacă se dorește sau este necesar ca alt programator să poată lucra cu codul respectiv mai târziu.

Bibliografie

1. James A. Storer, *An Introduction to Data Structures and Algorithms*, Springer Verlag, 2001
2. T. H. Cormen, C. E. Leiserson, R. R. Rivest, *Introducere în algoritmi*, Editura Computer Libris Agora, 2000
3. ***, *colecția GInfo*
4. **http://www.liis.ro/~campion**
5. **http://www.aggregate.org/MAGIC/**
6. **http://www.topcoder.com** – *RoundTables*

Cosmin-Silvestru Negruşeri este student în anul III la Universitatea Babeş-Bolyai din Cluj-Napoca. El poate fi contactat prin e-mail la adresa **kosmin_2000@yahoo.com**.

KNOPPIX Linux

Silviu Niculiță

Dacă doriți să încercați Linux fără a fi nevoiți să vă repartiționați hard-diskul sau vreți să aveți cele mai bune aplicații de securitate adunate pe un singur CD boot-abil, atunci Knoppix Linux este o soluție posibilă, iar prețul este potrivit, pentru că este 0.

Având în vedere popularitatea crescândă a *Linux*-ului, mulți utilizatori doresc să îl încerce, dar vor să păstreze intact *Windows*-ul. Distribuția *Knoppix Linux* a fost creată acum câțiva ani, exact în acest scop. Ea permite novicilor să se "joace" cu *Linux*-ul fără a se atinge de *hard-disk*-urile lor, dar este foarte atractivă și pentru utilizatorii avansați, datorită numărului mare de unelte de securitate și recuperare a datelor incluse.

Cum începem?

Pentru a începe să folosiți *Knoppix*, aveți nevoie de o copie a distribuției. Aceasta poate fi descărcată gratuit de pe *site*-ul dezvoltatorului (**http://www.knopper.net/knoppix**), *Klaus Knopper*, sau, dacă aveți o conexiune lentă, poate fi achiziționată pe *CD* din alte surse, contra unei sume cuprinsă între 3$ și 5$. Dacă decideți să descărcați versiunea gratuită, vă veți alege cu un fișier *ISO* de 696MB pe *hard-disk*, care va trebui inscripționat pe un *CD* de 700MB. În acest articol vom face referire la *Knoppix Linux* versiunea 3.3.

Cerințe minime

Cerințele minime de sistem pentru a rula *Knoppix* sunt: *Intel* 486 sau echivalent, 16 *MB* de *RAM* pentru modul text, cel puțin 96MB pentru interfața grafică *KDE* (cel puțin 128MB pentru a rula și pachetul de aplicații *OpenOffice*), o unitate *CD-ROM boot*-abilă și o placă video compatibilă cu standardul *SVGA*.

Software inclus pe CD-ul Knoppix

Principalele aplicații pe care le puteți găsi pe un *CD Knoppix* sunt:

- interfață grafică standard *KDE V3.x* , *KOffice* și programul de navigare *Konqueror*;
- *OpenOffice 1.x*, versiunea *GPL* a bine-cunoscutului pachet de aplicații *StarOffice*;
- *X Multimedia System* (*xmms*) cu suport pentru diferite formate *video* și *audio*;
- aplicații pentru accesarea internetului;
- unelte de analiză și administrare a rețelelor;
- *Gnu Image Manipulation Program* (*GIMP*) versiunea 1.2;
- diverse programe de recuperare a datelor;
- multe limbaje de programare, unelte de dezvoltare a aplicațiilor (cum este cazul aplicației ***kdevelop***) și librării pentru programatori.

În total, *Knoppix* include peste 900 de pachete *software*, cuprinzând peste 2000 de programe executabile, utilitare și jocuri.

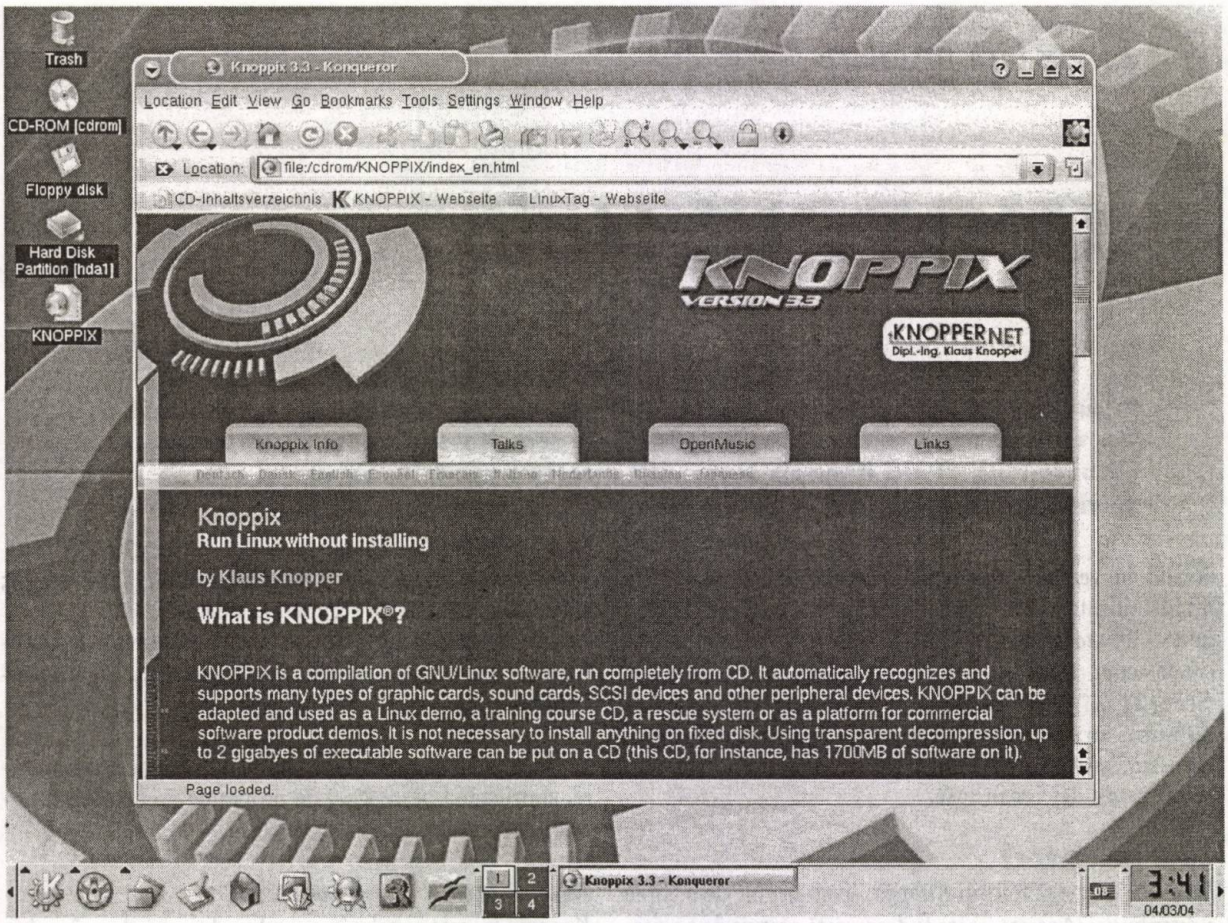
Instalare... Nu-i nevoie!

Knoppix nu este instalat în sensul tradițional al cuvântului. În schimb, el rulează direct de pe *CD*. Marea majoritate a sistemelor de operare cer utilizatorilor să creeze o partiție pe *hard-disk*, să o formateze, ca apoi sistemul de operare să își poată copia fișierele pe ea. *Knoppix* nu va scrie nimic pe *hard-disk*-ul calculatorului, decât dacă îi cereți explicit să facă asta. Pentru a rula *Knoppix*, tot ce trebuie să faceți este să introduceți *CD*-ul, și să porniți calculatorul.

Dacă aveți o cantitate mare de memorie *RAM* (cel puțin 512 *MB*), puteți tasta `"knoppix toram"`, instrucțiune care va copia conținutul întregului *CD* în memoria *RAM*. Această operațiune va dura mai multe minute, dar sporul de performanță va fi dramatic. În mod normal, când o aplicație este deschisă în *Knoppix*, aceasta este decompresată, și apoi încărcată în memorie. Această operațiune durează și creează utilizatorilor impresia că *Knoppix Linux* este lent. Dacă veți folosi comanda `"knoppix toram"`, atunci *Knoppix* va rula mult mai rapid, deoarece aplicațiile sunt decompresate direct în *RAM*. Este posibil să instalați *Knoppix* pe *hard-disk* (găsiți instrucțiuni complete aici: **http://www.knoppix.net/docs/index.php/FaqInstall**), dar dacă tot intenționați să vă bateți capul cu o instalare clasică, puteți instala o distribuție "adevărată" de *Linux*, cum ar fi *Red Hat Fedora* sau *Mandrake*.

Detectarea componentelor hardware

Knoppix beneficiază de capabilități foarte bune de detectare a *hardware*-ului. Detectează fără probleme marea ma-



Knoppix Linux

joritate a plăcilor video. *Knoppix*-ul este de asemenea o modalitate bună de a testa compatibilitatea *hardware*-ului pe care îl deținеți cu *Linux*. Dacă *Knoppix* detectează o componentă, atunci și celelalte distribuții o vor face.

Interfața

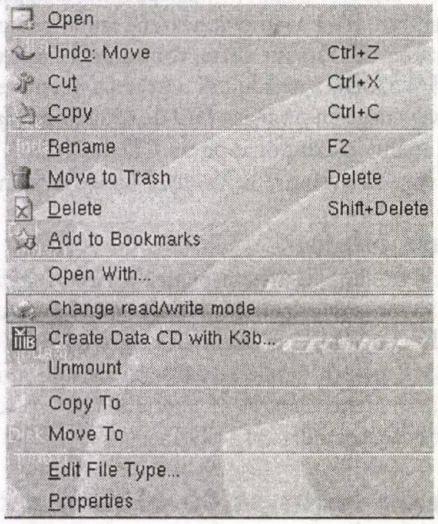
Utilizatorii de *Windows* se vor simți "ca acasă". După cum vedeți în imaginea de mai sus, interfața *Knoppix*-ului seamănă foarte mult cu cea a *Windows*-ului.

Toate partițiile și unitățile optice sunt listate în partea stângă a ecranului. Bara de aplicații din partea de jos a ecranului oferă legături la programe populare ca *OpenOffice*, *Mozilla* și consola *Linux*.

Sisteme de fișiere

Knoppix poate citi toate sistemele de fișiere *Windows* (*FAT*, *FAT32*, *NTFS*), cât și cele *Linux* (*ext2*, *ext3*, *ReiserFS*).În momentul în care *Knoppix boot*-ează, toate partițiile sunt "montate" automat cu drepturi de citire. Aceasta permite ștergerea sau modificarea accidentală a datelor de pe *hard-disk*.

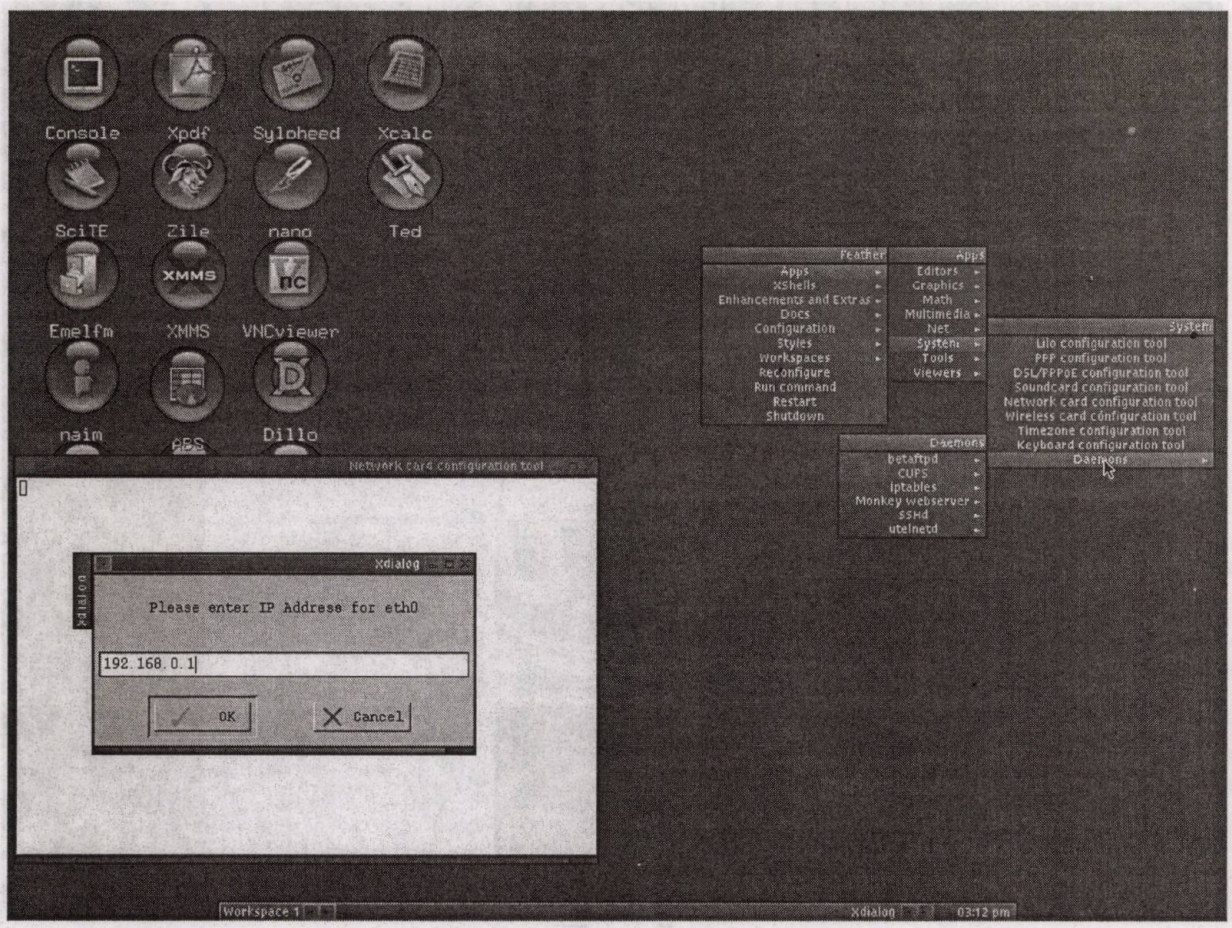
Dacă doriți, totuși, să scrieți pe *hard-disk*, tot ce trebuie să faceți este să dați un *click* dreapta pe pictograma asociată partiției respective, și apoi trebuie să selectați opțiunea "*Change read/write mode*". Mai simplu de atât nu se poate.



Knoppix 3.3 are totuși o limitare: încă nu poate scrie pe volume *NTFS*. Pentru a ne bucura de această funcție va trebui să așteptăm apariția versiunii 3.4.

Lucrul în rețea

Samba și *LinNeighborhood* vă vor ajuta să vă conectați cu *Knoppix* la rețeaua locală și să partajați fișiere pentru *Windows*. *LinNeighborhood* funcționează la fel ca "*My Network Places*" din *Windows*.



Feather Linux

Aplicații

OpenOffice 1.1, clona *Microsoft Office*, este inclusă în *Knoppix*. Încărcarea acestei aplicații durează foarte mult (peste 40 de secunde pe un sistem performant). Veți reuși totuși să creați un document folosind *OpenOffice*, să îl salvați pe un *hard-disk*, și să îl editați apoi sub *Windows*.

În afară de *OpenOffice*, pe *CD* veți găsi aproape 2000 de aplicații (unelte de inscripționat *CD*-uri, *player*-e audio și video, *browser*-e *web*, jocuri etc.). *Knoppix* include și emulatorul de *Windows Wine*, care permite rularea multor aplicații *Windows* sub *Linux*.

Personalizare

Knoppix poate fi personalizat sau poate suporta setările făcute de către utilizatori în funcție de pretențiile lui. Puteți îndepărta aplicațiile de care nu aveți nevoie și le puteți înlocui cu altele mai utile. Acest proces se numește *remasterizarea CD*-ului și a avut ca rezultat apariția a zeci de versiuni de *Knoppix*. Aici puteți găsi o listă completă a acestora:

http://www.knoppix.net/docs/index.php/KnoppixCustomizations

Un ghid de *remasterizare* a *Knoppix*-ului găsiți la : **http://www.knoppix.net/docs/index.php/KnoppixRemasteringHowto.**

Trebuie totuși să țineți cont de faptul că *CD*-ul original este optimizat pentru viteză (suferă un fel de "defragmentare"). Din acest motiv, *Knoppix*-ul original rulează mai repede decât majoritatea clonelor.

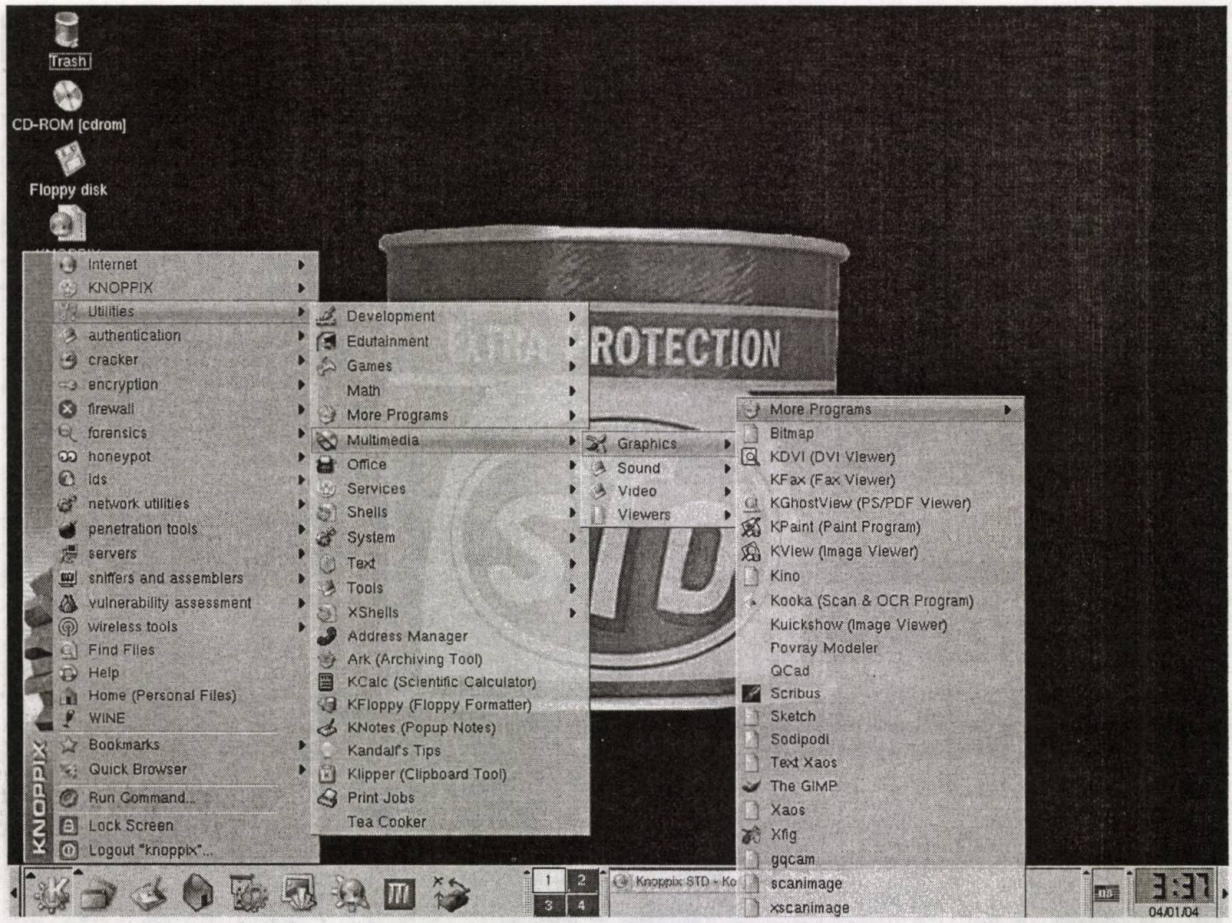
Clone Knoppix

Feather Linux (**http://featherlinux.berlios.de**)

Această distribuție de *Linux* ocupă doar 64MB. Exact, numai 64MB. Când accesați *Feather Linux* aveți câteva opțiuni legate de tipul de periferice folosit și rezoluția monitorului. În versiunile mai vechi erați "abandonați" în fața unei console și dacă doreați să lucrați cu interfață grafică (singura disponibilă este *Blackbox*), trebuia să tastați comanda `startx`. Acum interfața grafică a *Feather Linux* pornește implicit. Dacă vă puteți descurca cu *desktop*-ul *Windows*, *KDE* sau *Gnome*, s-ar putea să vă sperie puțin aspectul cu adevărat minimalist al *desktop*-ului *Feather Linux*. Sunt doar câteva aplicații... și cam atât. *Feather Linux* nu este recomandat novicilor. Pentru un programator, sau un utilizator avansat, care are nevoie de o distribuție "zveltă", care poate fi lansată imediat, *Feather Linux* este ideal.

Knoppix-STD (**http://www.knoppix-std.org**)

Knoppix-STD (*Security Tools Distribution*) conține foarte multe unelte de securitate a rețelelor. Această distribuție poate scrie pe volume *NTFS*. *Knoppix-STD* ar trebui să facă parte din arsenalul fiecărui administrator de rețea, în-



Knoppix-STD

trucât este o unealtă foarte bună pentru testarea securității unei rețele.

Salvarea setărilor

O altă caracteristică foarte importantă a *Knoppix*-ului este posibilitatea de a salva fișierele de configurare pe diverse dispozitive de stocare și de a le refolosi ulterior. Asta înseamnă că la *reboot* nu veți pierde modificările făcute. Pentru a salva modificările, mergeți la *K-menu* > *Knoppix* > *Configuration*, și alegeți opțiunea "*Save config*". Salvarea poate fi făcută pe o dischetă sau pe o partiție a *hard-disk*-ului.

Pentru a restaura o configurație salvată, la consola care afișează `boot:`, tastați `knoppix myconfig = scan` și ar trebui să găsească setările salvate și să le încarce.

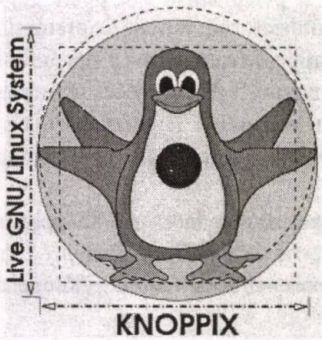
Concluzii

De la începători și până la cei mai avansați utilizatori de *Linux*, *Knoppix* este o unealtă foarte utilă. Este ușor de folosit și plină de aplicații. În plus, cu excepția costului inscripționării *CD*-ului, *Knoppix* este absolut gratuit. *Knoppix* este cu siguranță unul dintre cele mai bune produse ale comunității *open-source*. Dacă abia ați început să învățați *Linux*, puteți folosi *Knoppix* pentru a exersa fără a risca să vă dați calculatorul peste cap. Dacă sunteți un administrator de rețea (sau un *hacker*), *Knoppix* oferă o multitudine de aplicații de securitate. Acest *CD*, în mâinile unei persoane pricepute, poate provoca pagube serioase.

Singurele dezavantaje ale *Knoppix*-ului ar fi timpul mare de încărcare al aplicațiilor complexe (*OpenOffice*) și relativa lipsă de documentație. Aceasta este însă o caracteristică generală a comunității *Linux*.

Informații suplimentare legate de această distribuție a sistemului de operare *Knoppix* puteți găsi la adresa **http://www.knoppix.net**;

Silviu Niculiță este student în anul I la Universitatea Babeș-Bolyai din Cluj-Napoca. El poate fi contactat prin e-mail la adresa **silviun@gmx.net**.



Pagini WEB cu PHP 5

Mihai Scorțaru, Claudiu Soroiu

Acest articol este ultimul din seria destinată modului de utilizare a limbajului PHP 4 și interacțiunea sa cu alte medii. De data aceasta nu vom mai prezenta funcții și interacțiuni, ci vom prezenta îmbunătățirile care au fost aduse de versiunea 5 a interpretorului PHP.

Noutăți

Cele mai multe îmbunătățiri care au fost aduse limbajului *PHP* sunt legate de paradigma programării orientate obiect.

În cadrul acestei versiuni au fost rescrise complet rutinele de manipulare a obiectelor din cadrul *script*-urilor *PHP* astfel încât codul să se execute mai eficient și să ofere mai multe facilități programatorilor.

În versiunile precedente versiunii 5 a interpretorului, obiectele erau tratate ca tipuri primitive de date. Astfel, în momentul atașării unei valori de tip obiect unei variabile, obiectul era mai întâi copiat în zona de memorie corespunzătoare variabilei respective. În această versiune obiectele sunt identificate prin referințe, similar limbajului *Java*.

De exemplu, dacă avem următorul cod *PHP*:

```
class Complex{
  var $re;
  var $im;

  function Complex($re = 0, $im = 0){
    $this->setRe($re);
    $this->setIm($im);
  }

  function getRe(){
    return $this->re;
  }
  function setRe($re){
    $this->re = $re;
  }
  function getIm(){
    return $this->im;
  }
  function setIm($im){
    $this->im = $im;
  }
}

function changeValue($complex, $re, $im){
  $complex->setRe($re);
  $complex->setIm($im);
}

$complex = new Complex(1, 0.5);
changeValue($complex, 2, 5);
echo $complex->getRe();
```

în urma executării ultimei linii de cod se va obține valoarea 1 în locul valorii 2, deoarece, în momentul apelării funcției `changeValue`, se creează o copie a obiectului `$complex` și toate modificările se vor efectua asupra copiei fără a fi observate din exteriorul funcției.

Această eroare de programare poate fi reparată folosind transmiterea prin referință a obiectului `$complex` din cadrul funcției, folosind de mai multe ori operatorul `&`.

În *PHP* 5 obiectele sunt transmise prin referință tot timpul.

În continuare vom prezenta modificările aduse modelului obiect de versiunea 5 a interpretorului *PHP*.

Modificatorii de acces public, private și protected

În noua versiune a interpretorului *PHP* au fost introduse cuvintele cheie **`public`**, **`private`** și **`protected`**. Cei care au folosit limbaje de programare ca *C++* sau *Java* cunosc semnificația acestor cuvinte cheie în programarea orientată obiect.

Aceste cuvinte cheie sunt modificatori de acces la metodele și proprietățile unor clase.

Dacă în cadrul declarației unei clase, o metodă sau proprietate este precedată de cuvântul cheie **public**, atunci acea metodă va putea fi accesată din exteriorul declarației clasei.

Dacă în cadrul declarației unei clase, o metodă sau proprietate este precedată de cuvântul cheie **protected**, atunci acea metodă va putea fi accesată doar în cadrul declarației clasei curente și a claselor derivate din aceasta.

Dacă în cadrul declarației unei clase, o metodă sau proprietate este precedată de cuvântul cheie **private**, atunci acea metodă nu va putea fi accesată din exteriorul declarației clasei curente.

Dacă pentru un membru al unei clase nu este specificat nici un modificator de acces, acel membru va avea implicit modificatorul **public**.

În continuare vă vom prezenta un exemplu de utilizare corectă a acestor modificatori de acces la membrii claselor.

```
class Punct{
  private $x;
  private $y;

  public function Punct($x,$y){
    setXCoord($x);
    setXCoord($y);
  }

  function setXCoord($x){
    $this->x = $x;
  }
  function getXCoord(){
    return $this->x;
  }
  function setYCoord($y){
    $this->y = $y;
  }
  function getYCoord(){
    return $this->y;
  }

  //funcția următoare este corectă deoarece $punct are tipul
  //Punct
  protected function setLocation($punct){
    $this->x = $punct->x;
    $this->y = $punct->y;
  }
}

$punct = new Punct(1, 0.5);
echo $punct->x;                    //nu afișează nimic
$punctnou = new Punct(3, 2);
$punct->setLocation($punctnou); //nu face nimic
$punct->setXCoord(3);
$punct->setXCoord(5);
echo $punct->x." ".$punct->y;
```

Modificatorul final

În limbajul *PHP* a fost introdus cuvântul cheie **final**. Acest cuvânt cheie poate fi folosit în cadrul declarării de membri ai unei clase sau în cadrul declarării clasei.

Dacă un membru este declarat ca fiind **final**, atunci acel membru al clasei nu va putea fi suprascris (nu i se va putea modifica funcționalitatea) în cadrul claselor derivate.

Dacă o clasă este declarată ca fiind **final**, atunci nu se pot crea clase derivate din aceasta.

Poate vă întrebați care este utilitatea modificatorului **final**. Modificatorul **final** indică programatorului că elementul care are această proprietate reprezintă o operație sau structură *atomică* și o structură atomică nu mai poate fi divizată.

De exemplu, la nivel abstract, un punct este o structură atomică. În momentul în care acel punct trebuie desenat pe o suprafață de desen, atunci nu mai este o operație atomică deoarece poate fi desenat folosind mai multe culori.

În continuare vă prezentăm un exemplu de utilizare a acestui modificator.

```
final class ClasaAtom{
  function ClasaAtom(){
  }
}

//declarația următoare nu este validă
class ExtindeAtom extends ClasaAtom{
}

class Factory{
  function Factory(){
  }

  //metoda următoare nu poate fi suprascrisă de clasele derivate
  //din clasa Factory
  final function createInstance($a){
    ...
  }
}
```

Constructori

În versiunea 5 a interpretorului *PHP* au fost unificați constructorii. De această dată, pentru a construi un obiect, trebuie implementată metoda `__construct()`.

Clasele pe care le-am prezentat anterior în cadrul articolului sunt corecte pentru noul interpretor *PHP*, deoarece s-a păstrat compatibilitatea cu versiunea 4, astfel că dacă în cadrul definiției unei clase interpretorul nu găsește metoda `__construct()` atunci acesta va căuta o metodă a cărui nume este identic cu cel al clasei și o va folosi pe post de constructor.

De exemplu, clasa **Punct**, ca să fie compatibilă numai cu limbajul PHP 5, ar trebui implementată astfel:

```
class Punct{
  private $x;
  private $y;
```

```
  function __construct($x,$y){
    setXCoord($x);
    setXCoord($y);
  }
...
```

Am omis din acest exemplu celelalte metode ale clasei `Punct` deoarece numai constructorul este important în acest moment.

În *PHP* în momentul construirii unui obiect, nu este apelat constructorul clasei părinte, acesta trebuind apelat, dacă este cazul, folosind instrucțiunea:

```
parent::__construct().
```

Destructori

În cadrul claselor definite în limbajul *PHP* 5 pot fi declarați destructori de clase.

Destructorii sunt utilizați în momentul distrugerii (eliberării memoriei alocate) unui obiect. Un obiect este distrus (eliminat din memorie) în momentul în care nu mai există referințe către el. Necesitatea destructorilor a apărut în momentul în care s-a pus problema transmiterii obiectelor prin referință și nu prin valoare cum se efectua transmiterea obiectelor în versiunile anterioare ale interpretorului.

Un destructor este dat de o metodă al cărui nume este `__destruct()` și care nu are parametri.

Ca și în cazul constructorilor, în momentul apelului unui destructor pentru o clasă nu este apelat automat destructorul părintelui, acesta trebuid apelat folosind instrucțiunea:

```
parent::__destruct().
```

În continuare vă prezentăm un exemplu de utilizare a destructorilor.

```
class Linie{
  private $capat1;
  private $capat2;

  function __construct($p1, $p2){
    $this->capat1 = $this->p1;
    $this->capat2 = $this->p2;
  }

  function __destruct($p1, $p2){
    $this->capat1 = null;
    $this->capat2 = null;
  }
}
```

Operatorul delete

Deși pare ciudat, există momente din timpul execuției unui script *PHP* în care anumite variabile, care nu mai sunt utilizate, să refere niște obiecte. Din această cauză obiectele respective nu sunt distruse. Pentru a evita astfel de situații a fost introdus operatorul **delete**. Acest operator apelează metoda **__destruct()** a obiectului care va fi distrus și eliberează memoria asociată acestuia.

În continuare vă prezentăm un exemplu de utilizare a acestui operator.

```
$punct1 = new Punct(1, 1);
$punct2 = new Punct(2, 2);
$linie = new Linie($punct1, $punct2);
$linie_noua = $linie;
delete $linie;
```

Modificatorul static

În limbajul *PHP*, ca de altfel și în alte limbaje moderne de programare există modificatorul **static**.

Haideți să vedem ce înseamnă faptul că un membru al unei clase a fost declarat **static**.

În momentul creării unui obiect, se alocă o zonă de memorie pentru obiectul nou creat. Pentru membrii statici ai unei clase se creează o altă zonă de memorie în care vor fi stocați. Astfel, toate obiectele de un anumit tip partajează aceeași zonă de memorie pentru membrii statici.

Se recomandă folosirea modificatorului **static** pentru membrii unei clase care sunt independenți de instanțele acesteia, adică tot timpul vor avea aceeași valoare în cadrul tuturor instanțelor.

Vă recomandăm să nu deduceți, din faptul că un membru are aceeași valoare în cadrul tuturor instanțelor unei clase, că acel membru este o constantă deoarece nu este adevărat, și aceasta din cauză că în timpul execuției unui *script* valoarea membrului respectiv se poate modifica, dar se va modifica în același fel în toate instanțele.

În versiunea 5 a limbajului *PHP* modificatorul static poate fi aplicat metodelor unei clase, iar proprietățile statice ale unei clase pot fi acum inițializate în momentul declarării lor.

Membrii statici ai unei clase nu pot fi folosiți prin apel indirect (folosirea operatorului ->) ci folosind construcția: `NumeClasă::numeMembru`.

În continuare vă prezentăm un exemplu de utilizare a modificatorului **static**.

```
class Locale{
  private static defaultLocation = "Romania";
  private location;

  function __construct($location = null){
    if ($location == null)
      setLocation(Locale::$defaultLocation);
    else
      setLocation($location);
  }

  function getDefaultLocation(){
    return Locale::$defaultLocation;
  }
  function setDefaultLocation($location){
    Locale::$defaultLocation = $location;
  }
```

```
  function getLocation(){
    return $this->location;
  }
  function setLocation($location){
    $this->location = $location;
  }
}
```

Clase și metode abstracte

În cadrul programării orientate obiect, pentru a reutiliza clase, se recomandă crearea de ierarhii de clase și clasele să fie specializate pe anumite operații în sensul că operațiile comune sau chiar identice ale unor clase să fie definite în cadrul unui părinte.

Există cazuri în care două sau mai multe clase care sunt similare din punctul de vedere al unor proprietăți și comportament, dar în cadrul unui posibil părinte nu pot fi definite toate funcțiile comune claselor. În acest caz, ar trebui definite în părinte metodele comune claselor chiar dacă implementarea lor depinde de clasele în care sunt construite.

Pentru a ușura munca programatorilor, a fost introdus conceptul de *clasă abstractă* și *metodă abstractă*. Acest concept se regăsește și în *PHP* 5 și se poate profita de el prin utilizarea cuvântului cheie **abstract** înaintea declarației unei clase sau a unei metode.

O metodă este abstractă dacă îi lipsește implementarea.

O clasă este abstractă dacă nu a fost construită complet sau conține cel puțin o metodă abstractă.

Clasele abstracte nu pot fi instațiate deoarece în cadrul implementării anumitor metode din cadrul unei clase abstracte se pot apela metode care au fost declarate ca fiind abstracte.

Din punct de vedere logic o clasă nu poate fi declarată ca fiind **final** și **abstract** în același timp deoarece aceasta nu ar putea fi utilizată deloc în programare.

În continuare vă prezentăm un exemplu de utilizare al acestui cuvânt cheie **abstract**:

```
abstract class CoadaDePrioritati{
  abstract public function push($obj);
  public function pop(){
    $obj = $this->peek();
    $this->remove($obj);
    return $obj;
  }
  abstract public function peek();
  abstract protected function remove($obj);
  abstract public function isEmpty();
}

final class Heap extends CoadaDePrioritati{
  public function push($obj){
    ...                    //aici urmează implementarea
  }
  public function peek(){
    ...                    //aici urmează implementarea
  }
  public function remove($obj){
    ...                    //aici urmează implementarea
  }
  public function isEmpty(){
    ...                    //aici urmează implementarea
  }
}
```

Interfețe

În limbajul *PHP* 5 a fost introdus conceptul de interfață. O *interfață* este o clasă abstractă ai cărei membri sunt publici și care nu are nici o metodă implementată. De fapt, o interfață este un schelet pentru clasele care o vor implementa.

În *PHP* 5 o interfață se creează asemenea unei clase cu excepția faptului că metodele pot fi implementate, iar în locul cuvântului cheie **class** trebuie utilizat cuvântul cheie **interface**.

Spre deosebire de cazul în care o clasă este derivată din altă clasă și pentru a indica acest lucru, se folosește cuvântul cheie **extends**. În cazul în care o clasă implementează metodele prezente într-o interfață se va folosi cuvântul cheie **implements**, urmat de o listă de interfețe separate între ele prin virgulă.

În continuare vom prezenta un exemplu de utilizare a interfețelor în *PHP* 5.

```
interface Lista{
  public function add($obj);
  public function get($i);
  public function set($obj,$i);
  public function remove($i);
  public function size();
}

class Vector implements Lista{
  private $vector;

  public function __contruct(){
    $this->vector = array();
  }

  public function add($obj){
    $this->vector[] = $obj;
  }
  public function get($i){
    return $this->vector[$i];
  }
  public function set($obj,$i){
    if ($i <= $this->size)
      $this->vector[$i] = $obj;
  }
  public function remove($i){
    for($j = $i; $j < $this->size()-1; $j++)
      $this->vector[$j] = $this->vector[$j+1];
    array_pop($this->vector);
  }
```

```
  public function size(){
    return sizeof($this->vector);
  }
}
```

Operatorul instanceof

Acest operator este prezent și în limbajul *Java* și funcționalitatea sa este echivalentă cu cea a funcției `is_a()`. Acest operator este binar și are ca operatori un obiect (în partea stângă) și un nume de clasă sau interfață (în partea dreaptă) și efectul folosirii lui este valoarea logică **TRUE** dacă obiectul este o instanță a clasei sau este derivat din aceasta.

În continuare vă prezentăm un exemplu de utilizare a acestui operator.

```
$a = new Punct(1, 1);
if ($a instanceof Punct)
  echo "a este instanță a clasei Punct";
```

Clonarea explicită

Datorită faptului că în *PHP* 5 obiectele sunt transmise prin referință și nu prin valoare așa cum se transmiteau în versiunile anterioare, a apărut problema replicării obiectelor.

Pentru aceasta s-a introdus metoda `__clone()`. Pe lângă faptul că această metodă se implementează în cazul în care se dorește replicarea unui obiect, această replicare este controlată de programator și se poate face optim în funcție de fiecare clasă în parte.

În continuare vă vom prezenta un exemplu de replicare a unui obiect.

```
class Punct{
  public $x,$y;

  public function __construct(){
    $this->x = 0;
    $this->y = 0;
  }

  public function __clone(){
    $res = new Punct();
    $res->x = $this->x;
    $res->y = $this->y;
    return res;
  }
}

$p1 = new Punct();
$p1->x = 4;
$p1->y = 5;
$copie_p1 = $p1->__clone();
```

Metoda __toString()

În *PHP* 5 există posibilitatea ca orice obiect să aibă o reprezentare textuală. Dacă un obiect se folosește în operații cu șiruri de caractere, atunci se va folosi o reprezentare textuală a acestuia. În *PHP* 5, prin implementarea metodei `__toString()` programatorul poate controla modul în care un obiect va fi reprezentat sub forma unui șir de caractere.

În continuare vă prezentăm modul de utilizare a metodei `__toString()`.

```
class Punct{
  public $x,$y;
  public function __construct(){
    $this->x = 0;
    $this->y = 0;
  }

  public function __toString(){
    return "(".$this->x.", ".$this->y.")";
  }
}

$p1 = new Punct();
$p1->x = 4;
$p1->y = 5;
echo $p1;                    //se va tipări (4, 5)
```

Constante în interiorul claselor

În această versiune a limbajului *PHP* 5 se pot defini constante care să fie membri ai unor clase.

Constantele se declară similar proprietăților, dar sunt precedate de cuvântul cheie **const** și trebuie să fie inițializate la declarare.

În continuare vă prezentăm un exemplu de utilizare a constantelor.

```
class Clasa{
  const constant = "constant";
}

echo "Clasa::constant = ".Clasa::constant."\n";
```

Dereferențierea obiectelor

În versiunile precedente ale limbajului *PHP* nu erau posibile apeluri de forma `$obiect->metoda1()->metoda2()` în cazul în care `$obiect->metoda1()` returnează un obiect care are metoda `metoda2()`.

În *PHP* 5 acest lucru a fost făcut posibil prin dereferențierea obiectelor rezultate în urma apelurilor de metode și funcții.

Tipuri de date pentru parametri de tip obiect

În cadrul semnăturilor metodelor prezente în definiția unei clase, în fața parametrilor de tip obiect ale acestora se poate preciza tipul clasei pe care trebuie să îl aibă.

În momentul execuției codului *PHP*, dacă un obiect transmis ca parametru unei metode nu are tipul cerut, atunci este generată o eroare.

Nu pot fi impuse restricții asupra tipurilor variabilelor care au tipuri primitive (întreg, șir de caractere, real, vector etc.).

În continuare vă prezentăm un exemplu de utilizare a acestei facilități oferite de limbajul *PHP* 5.

```
class Linie{
  private $capat1;
  private $capat2;

  function __construct(Punct $p1, Punct $p2){
    $this->capat1 = $this->p1;
    $this->capat2 = $this->p2;
  }
```

Moştenirea multiplă

În *PHP* 5 a fost făcută posibilă moştenirea multiplă. O clasă acum poate fi derivată din una sau mai multe clase, deci poate să aibă mai mulţi părinţi. Acest lucru a afectat modul de funcţionare al funcţiei `get_parent_class()`.

Pentru a crea o clasă care să fie derivată din mai multe clase este nevoie ca după cuvântul cheie **extends** să se găsească lista numelor claselor moştenite separate între ele prin virgulă. Pentru a crea o clasă care are mai mulţi părinţi sintaxa este următoarea:

```
class Fiu extends Parinte1, Parinte2, ... {
  ...
};
```

Iteratori

În această versiune a limbajului *PHP* pot fi definite clase care enumeră elementele altei clase. Aceste clase se numesc *iteratori*.

Obiectele se pot folosi în cadrul ciclurilor repetitive **foreach** la fel ca şi tablourile unidimensionale create cu funcţia `array()` cu condiţia ca acestea să implementeze interfaţa `Traversable` care nu conţine nici o metodă şi nici o proprietate. Pentru obiecte cheile sunt reprezentate de numele proprietăţilor publice, iar valorile cheilor sunt chiar valorile proprietăţilor respective.

În continuare vă prezentăm un exemplu de enumerare a proprietăţilor unei clase:

```
class Punct2D implements Traversable{
  public $x = 4;
  public $y = 5;
}

$obj = new Punct2D();
foreach ($obj as $nume => $valoare){
  echo $nume."=".$valoare."\n";
}
```

Există posibilitatea ca o clasă să implementeze interfaţa `IteratorAggregate` şi să definească metoda `getIterator()` care trebuie să returneze un obiect care implementează interfaţa `Iterator`. Cu ajutorul acestor două interfeţe se specifică interpretorului *PHP* modul în care se va face enumerarea elementelor.

Interfaţa `Iterator` are următoarele metode:

- `rewind()` - trebuie să realizeze derularea la primul element al unui obiect (de exemplu, primul element al unui vector);
- `current()` - trebuie să returneze elementul curent;
- `key()` - trebuie să returneze cheia elementului curent;
- `next()` - trebuie să realizeze trecerea la următorul element dintr-o posibilă listă de elemente;
- `hasMore()` - trebuie sa returneze o valoare logică care indică dacă mai sunt elemente de enumerat sau nu.

În continuare vă prezentăm un exemplu de utilizare a acestei modalităţi de enumerare a elementelor unei clase.

```
class IteratorElemente implements Iterator{
  private $obj;
  private $nr;

  function __construct($obj){
    $this->obj = $obj;
  }
  function rewind(){
      $this->nr = 0;
  }
  function hasMore(){
    return $this->num < sizeof($this->obj);
  }
  function key(){
     return $this->nr;
  }
  function current(){
    return $this->obj[$this->nr];
  }
  function next(){
    $this->num++;
  }
}

class Obiect implements IteratorAggregate{
  function getIterator(){
    $a = array(1,2,3,4,5,"a","c");
    return new IteratorElemente($a);
  }
}

$obj = new Obiect;
foreach($obj as $cheie => $valoare){
  echo $cheie."=".$valoare."\n";
}
```

Supraîncărcarea accesului la membrii unei clase

O altă facilitate interesantă a limbajului *PHP* este aceea că oferă supraîncărcarea membrilor unei clase în sensul că se poate realiza accesul la diferite metode şi proprietăţi ale clasei care nu există în momentul creării unui obiect ci pot apărea pe parcursul execuţiei codului sau modificarea / aflarea valorilor unor proprietăţi implică şi alte calcule.

Dacă în cadrul unui obiect este implementată metoda `__get()` care primeşte ca parametru numele unei proprietăţi, atunci interpretorul *PHP* apelează la această metodă pentru a afla valoarea proprietăţii şi în acest fel se poate

controla modul în care valoarea unei proprietăți va fi returnată.

Dacă în cadrul unui obiect este implementată metoda `__set()` care primește ca parametri numele unei proprietăți și o valoare, atunci interpretorul *PHP* apelează la această metodă pentru a seta valoarea proprietății și în acest fel se poate controla modul în care valoarea unei proprietăți se modifică.

Dacă în cadrul unui obiect este implementată metoda `__call()` care primește ca parametri numele unei proprietăți și o listă de argumente, atunci interpretorul *PHP* apelează la această metodă în momentul apelului unei metode a obiectului respectiv, și în acest fel se poate controla modul și condițiile de execuție ale metodelor.

În continuare vă prezentăm un exemplu de utilizare a acestor facilități oferite de limbajul *PHP*.

```
class Setter{
  public $n;
  public $x = array("a" => 1, "b" => 2, "c" => 3);

  function __get($nm){
    echo "Returnare [$nm]\n";
    if (isset($this->x[$nm])) {
      $r = $this->x[$nm];
      echo "Am gasit proprietatea!\n";
      return $r;
    } else {
      echo "Nu am gasit proprietatea!\n";
  }
}

  function __set($nm, $val){
    echo "Setez valoarea pentru [$nm]\n";
    if (isset($this->x[$nm])) {
      $this->x[$nm] = $val;
      echo "Setare reusita!\n";
    } else {
      echo "Setare esuata!\n";
    }
  }
}

class Caller{
  var $x = array(1, 2, 3);

  function __call($m, $a){
    echo "A fost apelată metoda $m:\n";
    var_dump($a);
    return $this->x;
  }
}

$obj = new Setter();
$obj->n = 1;
$obj->a = 100;
$obj->a++;
$obj->z++;
var_dump($obj);

$foo = new Caller();
$a = $foo->test(1, "2", 3.4, true);
var_dump($a);
```

Alte elemente noi

Funcția __autoload()

Pentru a ușura munca programatorilor și codul *PHP* să nu conțină prea multe directive **include**, interpretorul *PHP* 5, în momentul în care găsește un nume de clasă care nu a fost încărcată în memorie, apelează funcția `__autoload()` dacă este definită. Această funcție este apelată cu un singur parametru de tip șir de caractere care reprezintă numele clasei care nu a fost găsită în memorie, deoarece se obișnuiește ca fiecare clasă să fie definită într-un script *PHP*.

În continuare vă vom prezenta un exemplu de utilizare a funcției `__autoload()`.

```
function __autoload($className){
  include_once $className.".php";
}

$object = new Punct;  //se va încărca fișierul Punct.php
```

Tratarea excepțiilor

În *PHP* 5 a fost introdus un mecanism de tratare a excepțiilor similar celui din *Java*, singura diferență fiind aceea că nu a fost implementată clauza **finally**.

Pentru a putea intercepționa o excepție care este posibil să fie generată de un bloc de cod, atunci acel bloc trebuie să se afle în interiorul unei clauze **try**. Clauza **try** poate fi urmată de una sau mai multe clauze **catch**. O clauză **catch** poate captura numai un anumit tip de excepție.

Tratarea excepțiilor se face în felul următor:

```
try{
  ... //secvență de cod
}
catch (TipExceptie1 exceptie1){
  ... //tratare TipExceptie1 reprezentată de exceptie1
}
catch (TipExceptie2 exceptie2){
  ... //tratare TipExceptie2 reprezentată de exceptie2
}
...
catch (TipExceptieN exceptieN){
  ... //tratare TipExceptieN reprezentată de exceptieN
}
```

unde `exceptie1`, `exceptie2`, ..., `exceptieN` sunt obiecte care au tipurile `TipExceptie1`, `TipExceptie2`, ..., respectiv `exceptieN`.

O excepție este reprezentată printr-o clasă definită de utilizator sau predefinită. În interiorul unui bloc de cod, o excepție este generată prin intermediul clauzei **throw** care este urmată de un obiect.

O excepție este *aruncată* la primul nivel care se află în interiorul unei clauze **try** și se verifică în ordine clauzele **catch** până în momentul în care tipul obiectului care constituie excepția este cel prezent în clauză (sau derivat din acesta). Dacă nici un tip de obiect prezent într-o clauză **catch** a unui bloc **try** nu se potrivește cu cel al excepției generate, atunci excepția generată este aruncată la nivelul următor.

Dacă s-a ajuns la cel mai de sus nivel, atunci excepția este tratată de interpretorul *PHP*.

În continuare vă prezentăm un exemplu de tratare a excepțiilor în limbajul *PHP* 5.

```
class ExceptieProprie{
  function __construct($exception){
    $this->exception = $exception;
  }

  function Display(){
    print "ExceptieProprie: $this->exception\n";
  }
}
class ExceptieProprieExt extends
                                ExceptieProprie{
  function __construct($exception){
    $this->exception = $exception;
  }

  function Display(){
    print "ExceptieProprie: $this->exception\n";
  }
}

try{
  throw new ExceptieProprieExt('Eroare');
}
catch (ExceptieProprie $exception){
  $exception->Display();
}
catch (Exception $exception){
  echo $exception;
}
```

Se recomandă ca excepțiile definite de utilizator să extindă clasa `Exception` deoarece aceasta oferă mai multe informații legate de generarea acestora.

Ciclul repetiv foreach cu referințe

În această versiune a interpretorului *PHP* 5 este posibil ca în cadrul ciclui repetitiv **foreach** sa apară referințe.

Astfel că în *PHP* 5 este posibilă execuția unui cod de forma:

```
foreach ($vector as &$valoare){
  if ($valoare === "NULL"){
    $value = null;
  }
}
```

Valori implicite pentru parametri transmiși prin referință

În limbajul *PHP* 5 pot fi declarate metode și funcții ai căror parametri care sunt transmiși prin referință pot avea valori implicite, și, în concluzie pot fi opționali.

În continuare vă prezentăm un exemplu de utilizare a acestei facilități.

```
function my_function(&$var = null){
  if ($var === null){
    echo "$var trebuie sa aiba o valoare";
}
```

Constanta __METHOD__

În limbajul *PHP* 5 a fost introdusă constanta `__METHOD__` a cărei valoare este dată de numele metodei în care constanta este folosită.

În continuare vă prezentăm un exemplu care ilustrează valoarea constantei `__METHOD__`.

```
class Clasa{
  function Afiseaza(){
    echo __FILE__.'('.__LINE__.')'.__METHOD__;
  }
}

function Test(){
  echo __FILE__.'('.__LINE__.')'.__METHOD__;
}

$obj = new Clasa();
$obj->Afiseaza();
Test();
```

În loc de încheiere...

În limbajul *PHP* 5 au fost introduse mult mai multe elemente decât cele prezentate în cadrul acestui articol.

Din păcate documentația disponibilă pe piață legată de acest limbaj este puțină și așteptăm și noi cu nerăbdare prezentări mai complete ale noilor facilități oferite de acest limbaj.

Versiunea 5 a limbajului *PHP* aduce cu ea biblioteci de funcții noi, dar mare parte din cele prezente și în versiunea 4 au fost optimizate din punct de vedere al vitezei de execuție a codului.

Bibliotecile nou introduse au fost realizate folosind facilitățile programării orientate obiect oferite de această versiune.

Mai mult, limbajul *PHP* 5 oferă suport pentru introspecție și reflexie, astfel că se pot afla foarte multe informații legate de clasele și membrii acestora și se pot afla informații legate de comentariile prezente în codul *PHP*.

Dacă sunteți interesați de această versiune a interpretorului *PHP* puteți să o descărcați de pe site-ul **www.php.net** și să testați voi înșivă câteva elemente din cele prezentate de-a lungul acestui articol, unele dintre ele fiind foarte interesante din foarte multe puncte de vedere.

Cel mai tare ghid IT
cele mai-noi știri
-noi produse
-tari teste
-bune soluții-hard
-soft
-internet
Din aprilie,
citește noul PC Magazine
DELL
Vrei să ști
PC MAGAZINE
ROMÂNIA
www.pcmagazine.ro
SOLUȚII
Tempus
Automatizarea completărilor în Excel
Crearea unui text animat
Despre MIDlet-uri și integrarea cu alte sisteme
INTERNET PRO
Securitatea Rețelei (5): atacurile bios
JSP: Java și mai aproape de noi
Transmisia prin fibra optică
GHIDUL TĂU ÎN LUMEA TEHNOLOGIEI
LG
Life's Good
Monitor LG Flatron LCD L1720P
Windows INCOMING

Publicarea acestei reviste este posibilă datorită companiilor de mai jos care sprijină învățământul românesc de informatică

SPONSORI DE PLATINĂ

FORTE COMPUTERS
Flexibility. Made in Romania

IBM®

sat
IT SOLUTIONS & SERVICES

hp®

SPONSOR DE AUR

CCT
CABINETUL CĂLIN TATOMIR

Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-266.516

GInfo

Octombrie 2004
Vol. 14/6

Lei 30.000

Gazeta de informatica

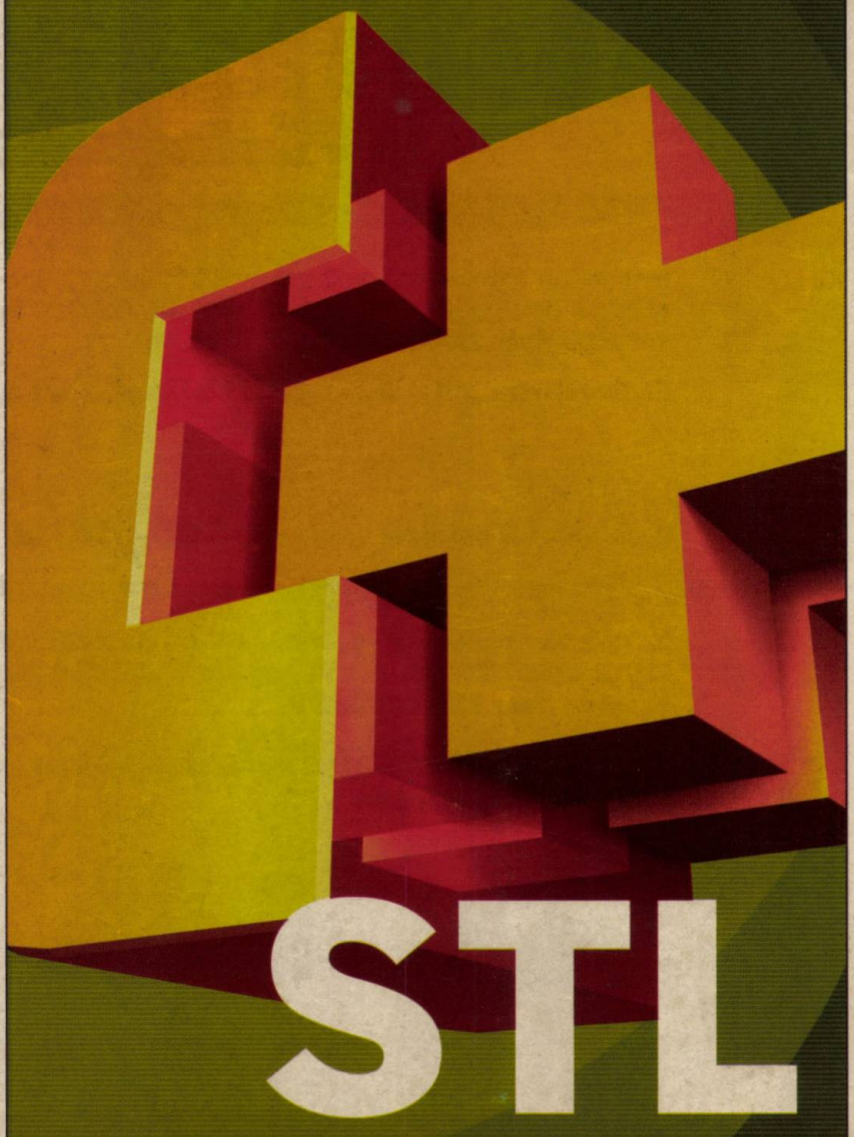
Revista viitorilor specialiști în tehnologia informației

Atena

din sumarul numărului:

Babel

STRUCTURI DE DATE



STL în concursuri

Plus

focus

Heap-uri de intervale: între min și max

info

IOI 2004: dupa 13 ani, din nou la Atena

concurs

25 septembrie: finala concursului GInfo

Fă-ți singur

UN CADOU DE SĂRBĂTORI!

Abonează-te

la PC Magazine România!

Încheie un abonament pe 2 ani la PC Magazine România
și în loc de: ~~74.000*~~ lei x 24 numere = 1.776.000 lei
vei plăti doar 1.440.000 lei!

Economisești astfel 336.000 lei și primești automat cadou:

un Webcam Canyon

de la ASBIS România în valoare de 465.000 lei

sau

Încheie un abonament pe 1 an la PC Magazine România
și în loc de: ~~74.000*~~ lei x 12 numere = 888.000 lei
vei plăti doar 720.000 lei!

Economisești astfel 168.000 lei și primești automat cadou unul din perifericele de mai jos:**

un mouse A4Tech
de la Python Systems, în valoare de
150.000 lei

o tastatură Genius
de la Python Systems, în valoare de
220.000 lei

un set de boxe A4Tech
de la Python Systems, în valoare de
190.000 lei

** vă rugăm să bifați pe talon cadoul preferat (o singură alegere)

sau

Încheie un abonament pe 6 luni la PC Magazine România și în loc de: ~~74.000*~~ lei x 6 numere = 444.000 lei vei plăti doar 360.000 lei!

Economisești astfel 84.000 lei

Pentru a vă abona vă rugăm să completați și să expediați pe adresa noastră talonul inserat în revistă sau să completați formularul de la adresa web ***www.pcmagazine.ro/abonamente***.

Oferta este valabilă în perioada 15 octombrie 2004 – 15 ianuarie 2005

*74.000 lei va fi prețul de vânzare pe exemplar începând cu 1 ianuarie 2005.

Vol. 14, Nr. 6 /2004

gazeta de informatică

Redacția:
Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. dr. *Clara Ionescu*
Redactor-șef:
Mihai Scorțaru
Redactor-șef adjunct:
Carmen Scorțaru
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Tudor Orha
Colaboratori:
Cosmin-Silvestru Negrușeri
Marius Andrei

Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
400091, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/592.422, 0265/266.516
Fax: 0265/266.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/262.359
Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
540014 Târgu-Mureș
Director general:
Romulus Maier
Director executiv:
Adrian Pop

Vânzare reclamă:
Mihai Bucuroiu
Denis Marioara
Marius Comărniceanu
Difuzare:
Florin Dinu
Florin Oprescu
E-mail: difuzare@buc.agora.ro
Tel: 021/330.92.82, int. 134
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Sumar

info.. PAG. 4

IOI 2004

CEOI 2004

BOI 2004

probleme.. PAG. 8

Olimpiada INTERNAȚIONALĂ de Informatică 2004

Olimpiada de INFORMATICĂ a EUROPEI CENTRALE 2004

Olimpiada BALCANICĂ de Informatică 2004

promo.. PAG. 23

PLEDOARIE pentru instruirea ASISTATĂ de CALCULATOR

concurs.. PAG. 25

Etapa finală

Bursele Agora

Ediția 2004 / 2005

Regulament

focus.. PAG. 34

Heap-uri de INTERVALE

Mihai SCORȚARU

babel.. PAG. 37

STL în concursuri

Marius ANDREI

serial.. PAG. 43

Modele de PROIECTARE

Tudor ORHA

GInfo încurajează publicarea de articole de către cititorii săi; în fiecare număr apar articole care nu sunt scrise de membrii redacției.

Dacă doriți ca numele dumneavoastră să fie inclus pe lista colaboratorilor revistei noastre, tot ce trebuie să faceți este să ne trimiteți un articol în vederea publicării.

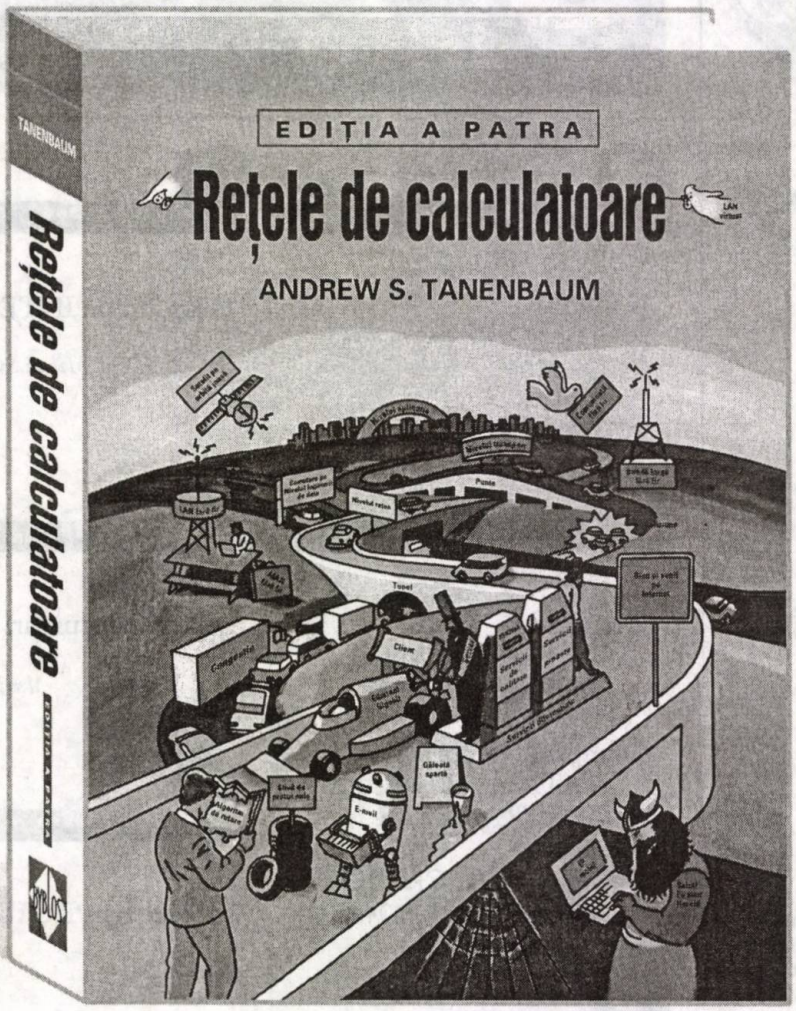
Singura condiție pe care trebuie să o îndeplinească articolul este ca tema să se încadreze în profilul revistei noastre.

Este recomandabil ca nivelul de accesibilitate să fie unul mediu și înțelegerea informațiilor prezentate să nu implice deținerea de cunoștințe avansate din alte domenii. Pentru mai multe detalii vizitați:

http://www.ginfo.ro/revista/publicare.shtml

Cine are carte, are parte!

... de înțelegerea rețelelor!



Cea mai bună introducere din lume în rețele de calculatoare
- complet actualizată pentru tehnologiile cheie de mâine -

Comand-o acum și dacă ești student primești și o reducere semnificativă!

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65; 0721-927.499

Un nou AN

Deși cu o întârziere destul de mare, aveți acum în mână un nou număr al revistei noastre. Poate v-ați speriat că nu veți mai putea citi prea curând o revistă ca a noastră, sau poate nu ați fost prea interesați de ceea ce se întâmplă cu GInfo. Cu toate acestea, sperăm că ne veți mai suporta încă un an și că nu veți fi dezamagiți de faptul că sunteți cititori ai singurei reviste tipărite din lume care se adresează elevilor și studenților care studiază informatica.

Ca în fiecare toamnă, *GInfo* revine! Nu vreau să vă plictisesc cu dificultățile inerente pe care le-am înfruntat, nu vreau nici să vă explic cât de greu ne este, nici câte sacrificii facem, etc. Important este că am reușit să revenim și *GInfo* va apărea pentru încă un an. Va fi ultimul? Nu se știe! Este posibil și este mai probabil decât înainte. Dar, vom încerca în continuare să vă oferim această revistă... În mod normal, în astfel de situații se face apel la ajutorul cititorilor dar, se pare că de această dată, nu mai depinde de dumneavoastră. Va trebui să ne descurcăm aproape singuri, să atragem sponsorii... Bineînțeles, faptul că o mare parte dintre dumneavoastră sunteți abonați, iar ceilalți cumpărați revista mai des sau mai rar ne ajută. Dar, pentru o astfel de revistă se pare că nu este suficient!

De obicei, un nou început înseamnă o mulțime de modificări. După cum puteți vedea pe prima pagină, structura echipei redacționale s-a modificat (pentru redactorul-șef adjunct s-a schimbat doar numele, persoana fiind aceeași! ☺), iar dacă răsfoiți revista veți observa că am încercat ca, în acest an, textul să fie puțin mai ușor de citit. Sperăm că imaginile care apar în articole vă vor ajuta să parcurgeți mai ușor conținutul acestora și să nu vă plictisiți citind, ca în anii trecuți, un șir aproape interminabil de cifre și litere.

Bineînțeles, trebuie să vă spun și ce ne propunem în noul an. Răspunsul este foarte simplu: *nimic special*. Vom avea o nouă ediție a concursului de programare *Bursele Agora* (regulamentul a fost modificat), vom prezenta în continuare enunțurile problemelor de la principalele concursuri de programare din țară și străinătate, în măsura posibilităților vom prezenta și soluțiile acestor probleme, vom avea articole referitoare la diferite subiecte (interesante, sperăm noi). În mare, revista va arăta cam la fel ca în anii anteriori.

Totuși, în acest an ne propunem să vă prezentăm și o altă față a programării, pe care nu o veți descoperi dacă tot ceea ce faceți este să rezolvați probleme. La majoritatea concursurilor ideea de bază este: *totul pentru performanță*. Suntem gata pentru orice compromis pentru a face ca programele noastre să funcționeze în continuare corect, dar să fie mai rapide.

În industria *software*, performanța, deși este un factor important, nu are rolul decisiv în momentul în care se proiectează o aplicație. Pentru a vă arăta această față am introdus un serial dedicat modelelor de proiectare utilizate cel mai frecvente. Pe parcursul întregului an, noul nostru coleg *Tudor Orha* vă va prezenta aceste elemente.

Deși membrii redacției *GInfo* nu mai au de câțiva ani un rol important în învățământul românesc de informatică, vom încerca, pe cât posibil, să vă ținem la curent cu ceea ce se întâmplă cu elevii și studenții care învață informatică, precum și cu profesorii lor.

Țin să mulțumesc acum sponsorilor care ne-au ajutat în anul precedent și celor care ne vor ajuta în noul an. Fără contribuțiile acestora, *GInfo* ar fi dispărut deja deoarece segmentul de piață căruia i se adresează nu este foarte larg, deci nu putem supraviețui deocamdată doar prin intermediul cititorilor noștri. Nu pot decât să le promit că vom face tot posibilul ca investițiile lor să aibă efect asupra nivelului învățământului românesc de informatică.

De asemenea, țin să mulțumesc cititorilor fideli care ne-au arătat că ceea ce facem noi nu este în zadar și care ne-au suportat de-a lungul anilor, indiferent cât de reușite sau nereușite au fost numerele revistei noastre.

Deși acest prim număr al toamnei 2004 a întârziat destul de mult și imediat se apropie vacanța de iarnă, nu pot să închei acest prim editorial al noului an fără să vă urez în numele redacției noastre un an școlar/universitar plin de satisfacții.

Mihai Scorțaru

IOI 2004

După 13 ani, Olimpiada Internațională de Informatică s-a întors la Atena. Competiția s-a desfășurat la numai câteva săptămâni după încheierea ediției din acest an a Jocurilor Olimpice. Între 11 și 18 septembrie, informaticienii au avut și ei propriul sat olimpic (pe care l-au numit Satul IOI).

Ediția din acest an s-a desfășurat sub auspiciile *Ministerului Educației* din *Grecia*, printre organizatori numărându-se *Societatea de Calculatoare* din *Grecia*, *Departamentul de Informatică și Telecomunicații* din cadrul *Universității Naționale* din *Atena*, precum și *Departamentul de Informatică* din cadrul *Institutului de Educație Tehnologică* din *Atena*.

La această competiție au participat concurenți din 80 de țări ale lumii, numărul total al participanților fiind 324.

Ei s-au confruntat pe parcursul a două zile de concurs, în fiecare dintre acestea ei trebuind să rezolve trei probleme pentru care au avut la dispoziție patru ore și treizeci de minute.

Mai multe informații referitoare la această ediție a concursului pot fi accesate pe *site*-ul oficial al competiției: **www.ioi2004.org**.

Dotări

În timpul concursului, fiecare concurent a avut la dispoziție câte un calculator cu următoarea configurație *hardware*:

- procesor *Intel Pentium* 4 cu frecvența de 3 GHz și memorie cache de 512 KB;
- memorie *DDR* de 512 MB;
- disc de 80 GB cu o viteză de 7200 rpm;
- monitor *TFT* cu diagonala de 17 inci;
- tastatură *Microsoft*;
- mouse optic *Microsoft*;
- unitate *DVD±RW*;
- unitate floppy de 3.5 inci.

Concurenților li s-a permis să utilizeze *CD*-uri pentru a-și păstra în siguranță datele.

Pe aceste calculatoare au fost instalate sistemele de operare *Microsoft Windows XP Professional* și *Red Hat Linux* 9.0.

Pentru sistemul de operare *Windows*, au fost instalate următoarele aplicații:

- *rhide* 1.5
- mediul de programare *FreePascal* 0.9.2
- compilatorul *FreePascal* 1.0.10
- compilatorul *gcc/gpp* 3.2.2
- utilitarul *gdb* 5.3
- utilitarul *make* 3.79.1
- utilitarul *emacs* 20.5
- utilitarul *ctags* 20.5

Pentru sistemul de operare *Linux*, concurenții au avut la dispoziție următoarele pachete *software*:

- *rhide* 1.5
- compilatorul *FreePascal* 1.0.10
- compilatorul *gcc/gpp* 3.2.2
- utilitarul *gdb* 5.3
- utilitarul *ddd* 3.3.1
- utilitarul *make* 3.79.1
- utilitarul *emacs* 21.2
- utilitarul *xemacs* 21.4
- utilitarul *vim* 6.1
- utilitarul *joe* 2.9.7
- utilitarul *ctags* 5.4
- manager-ul de fișiere *mc* 4.6.0
- programul de navigare *mozilla* 1.2
- programul de navigare *lynx* 2.8.5
- pachetul *Gnome* 2.2
- pachetul *KDE* 3.1

Programul

Pe lângă obișnuitele zile de concurs și ceremoniile de deschidere și închidere, programul ediției din acest an a inclus numeroase momente interesante cum ar fi vizita la celebrele *Acropole*.

De asemenea, organizatorii le-au oferit participanților diverse modalități agreabile pentru a-și petrece timpul liber.

Echipa României

Cei patru concurenți care au reprezentat țara noastră la această compe-

Primii trei clasați la IOI 2004

tiție au fost *Sorin Stancu-Mara*, *Silviu Gânceanu*, *Adrian Vladu* și *Dan-Leonard Crestez*.

La sfârșitul concursului, echipa *României* a reușit să obțină o medalie de aur, două de argint și una de bronz.

Cel mai bine clasat concurent român a fost *Silviu Gânceanu*, care s-a clasat pe poziția a 21-a și a obținut singura medalie de aur a *României*.

Dan-Leonard Crestez
410 puncte
argint

Următorul clasat dintre români a fost *Dan-Leonard Crestez*, care a ocupat locul al 44-lea și a obținut o medalie de argint.

Tot o medalie de argint a obținut și *Adrian Vladu*, care a fost al 55-lea concurent clasat la IOI 2004.

Cel de-al patrulea concurent român, *Sorin Stancu-Mara*, s-a clasat pe poziția a 85-a și a obținut o medalie de bronz.

Câștigătorii

Concursul a fost câștigat de britanicul *Paul Jefferys* care a reușit să obțină 565 din maximul de 600 de puncte.

Pe locurile următoare s-au clasat concurenții chinezi *Weidong Hu* și *Tiancheng Lou* cu 560, respectiv 530 de puncte.

Sunt demne de remarcat performanțele echipelor *Chinei* și *Rusiei*; toți concurenții din aceste țări au reușit să obțină medalii de aur.

Sorin Stancu-Mara
345 de puncte
bronz

Suprinzătoare a fost clasarea pe locul al cincilea în clasamentul neoficial pe medalii a echipei *Croației*, care a reușit să obțină două medalii de aur.

Croații au fost devansați de polonezi și de americani, care au obținut tot două medalii de aur, dar au obținut o medalie de argint în plus față de croați.

Poziția a șasea a clasamentului a fost împărțită de echipele *Iranului* și *Bulgariei* care au obținut o medalie de aur și trei de argint.

În acest clasament, echipa *României* a ocupat cea de-a opta poziție.

Pe lângă țările amintite deja, medalii de aur au mai obținut concurenți din *Thailanda*, *Singapore*, *Africa de Sud*, *Sri Lanka*, *Coreea de Sud*, *Germania*, *Olanda* și *Italia*.

Silviu Gânceanu
455 de puncte
aur

În total, au fost acordate 26 de medalii de aur, 49 de argint și 71 de bronz.

Pentru a obține medalia de aur au fost necesare 445 de puncte, pentru o medalie de argint au fost suficiente 365 de puncte, iar pentru una de bronz 265 de puncte.

Rezultate

În cele ce urmează vă vom prezenta cele mai importante performanțe obținute de către participanți în cele două zile de concurs.

Prima zi

Nici unul dintre concurenți nu a reușit să obțină punctajul maxim în prima zi de concurs. Primii doi clasați în clasamentul final au avut cele mai bune rezultate și în prima zi a competiției.

În fruntea clasamentului primei zile li s-a alăturat americanul *Brian Jacokes*.

Cei trei au reușit să obțină câte 100 de puncte la una dintre probleme, câte 90 de puncte la o altă problemă și câte 80 de puncte la ultima dintre cele trei probleme.

Rezultatele elevilor români au fost foarte apropiate în prima zi, punctajele lor fiind cuprinse între 155 și 180 de puncte.

A doua zi

Situația a fost diferită în cea de-a doua zi a concursului, când patru dintre concurenți au reușit să obțină punctajul maxim.

Unul dintre ei a fost concurentul care, în final, s-a clasat al treilea. Lui i s-au alăturat polonezul *Bartlomiej Romanski*, chinezul *Rong Ge* și americanul *Alex Schwendner*.

O mulțime de alți concurenți au obținut punctaje semnificative. Cincisprezece dintre participanți au reușit să obțină 295 de puncte, în timp ce alți doi concurenți au acumulat 290 de puncte.

În această a doua zi rezultatele membrilor echipei noastre nu au mai fost la fel de apropiate, ele variind între 185 și 280 de puncte.

Comparație

Așa cum se poate observa studiind aceste punctaje, dificultatea problemelor din cea de-a doua zi a fost mai redusă.

Ca urmare, o mulțime de concurenți au reușit să obțină punctaje mari în această a doua zi.

Lupta pentru primul loc a fost adjudecată în această a doua zi, când concurentul clasat în final pe primul loc a obținut 295 de puncte, în timp ce următorul clasat a trebuit să se mulțumească cu 290 de puncte.

Adrian Vladu
385 de puncte
argint

Datorită faptului că cei doi au fost la egalitate în prima zi, diferența în clasamentul final a fost de numai 5 puncte din 600 posibile.

Se poate spune că această a doua zi a fost mai importantă, deoarece concurenții aveau șanse mai mari de a obține un număr mai mare de puncte.

IOI 2005

Următoarea ediție a *Olimpiadei Internaționale de Informatică* va avea loc la *Nowy S'cz*, în *Polonia*, între 18 și 25 august 2005.

Site-ul oficial al ediției 2005 a *Olimpiadei Internaționale de Informatică* este **www.ioi2005.pl**.

CEOI 2004

Ediția 2004 a Olimpiadei de Informatică a Europei Centrale a avut loc în orașul polonez Rzeszów între 12 și 18 iulie. A fost a doua ediție care s-a desfășurat în Polonia, după cea de acum șapte ani. Pentru polonezi, aceasta a fost o excelentă ocazie de a se pregăti pentru organizarea Olimpiadei Internaționale de Informatică din 2005.

Competiția din acest an s-a desfășurat la *Universitatea de Tehnologia Informației și Management* din *Rzeszów*, fiind organizată sub egida *Ministerului Educației* din *Polonia*.

La concurs au participat 40 de concurenți din opt țări: *Bosnia și Herțegovina*, *Cehia*, *Croația*, *Germania*, *Polonia*, *România*, *Slovacia* și *Ungaria*.

Au fost două zile de concurs, în care participanții au avut de rezolvat câte trei probleme, pe parcursul a cinci ore.

Mai multe informații sunt disponibile pe *site*-ul oficial al concursului: **www.oi.edu.pl/ceoi2004**.

Dotări

Calculatoarele puse la dispoziție de către organizatori au avut următoarea configurație *hardware*:

- procesor *Intel Pentium* 4 cu frecvența de 1,6 GHz
- memorie *DDR* de 256 MB;
- monitor *CRT*;
- mouse și tastatură.

Pe aceste calculatoare au fost instalate sistemele de operare *Microsoft Windows XP Professional* și *Linux Debian 'Sid'*.

Sorin Stancu-Mara
186 de puncte
bronz

Pentru sistemul de operare *Windows*, au fost instalate următoarele aplicații:

- mediul de programare *rhide*
- compilatorul *FreePascal* 1.0.10
- compilatorul *gcc* 3.3
- pachetul *DJGPP*.

Alexandru Mosoi
219 puncte
argint

Pentru sistemul de operare *Linux*, concurenții au avut la dispoziție următoarele pachete *software*:

- mediul de programare *xwpe*
- mediul de programare *rhide*
- compilatorul *FreePascal* 1.0.10
- compilatorul *gcc* 3.3
- utilitarul *ddd*
- utilitarul *emacs21*
- utilitarul *emacs21-el*
- utilitarul *vim*
- utilitarul *vim-gtk*
- utilitarul *joe*
- manager-ul de fișiere *mc*
- programul de navigare *mozilla*
- programul de navigare *mozilla-fireworks*
- pachetul *KDE* 3.2
- documentațiile *FreePascal* și *STL*.

Echipa României

La *CEOI* 2004, *România* a fost reprezentată de *Mircea-Adrian Digulescu*, *Alexandru Mosoi*, *Dan-Constantin Spătarel* și *Sorin Stancu-Mara*.

Ei au reușit să obțină două medalii de argint și două de bronz.

Dan Spătărel
200 de puncte
argint

Câștigătorii

Mircea Digulescu
137 de puncte
bronz

Concursul a fost câștigat de croatul *Luka Kalinovcic* care a reușit să obțină 463 de puncte din 600 posibile.

Pe locurile următoare s-au clasat polonezul *Filip Wolski* și croatul *Lovro Puzar* cu 357, respectiv 327 de puncte.

La fel ca și la *IOI*, croații au făcut o figură surprinzător de frumoasă, clasându-se pe poziția a doua în clasamentul neoficial pe medalii, fiind devansați doar de către echipa gazdă.

Croații nu au reușit să obțină nici o altă medalie pe lângă cele de aur, în timp ce concurenții polonezi au reușit să obțină două medalii de aur, una de argint și una de bronz.

Echipa *României* a ocupat cea de-a treia poziție în acest clasament. În total, au fost acordate patru medalii de aur, cinci de argint și zece de bronz.

Pentru a obține medalia de aur au fost necesare 299 de puncte, pentru o medalie de argint au fost suficiente 200 de puncte, iar pentru una de bronz 107 puncte.

CEOI 2005

Următoarea ediție a *Olimpiadei I de Informatică a Europei Centrale* va avea loc în *Croația*.

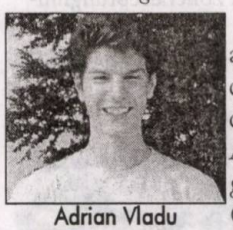
BOI 2004

Cu câteva zile înainte de CEOI, a avut loc și cel de-al doilea concurs internațional de informatică inițiat de țara noastră. Cea de-a XII-a ediție a Balcaniadei de Informatică a avut loc la Plovdiv, între 3 și 9 iulie și a fost a doua ediție desfășurată pe teritoriul Bulgariei, după cea de la Varna din 1995.

Balcaniada din acest an s-a desfășurat sub auspiciile prințului *Simeon Saxe-Coburg Gotha*, primul ministru al *Republicii Bulgaria*.

Numeroase instituții ale statului bulgar au colaborat la organizarea acestui important eveniment. Printre acestea s-au numărat *Ministerul Educației și Științei* din *Bulgaria*, *Agenția de Dezvoltare a Tehnologiei Informației și Comunicațiilor* din *Bulgaria*, guvernatorul provinciei *Plovdiv*, municipalitatea orașului *Plovdiv*, *Uniunea Matematicienilor Bulgari*, *Facultatea de Matematică și Informatică* din cadrul *Universității "Paisii Hilendarski"* din *Plovdiv*, *Inspectoratul Regional al Educației* din *Plovdiv*, *Liceul de Matematică "Academician Kiril Popov"* din *Plovdiv*, *Asociația pentru Tehnologiile Informației* din *Bulgaria* și *Asociația Companiilor Producătoare de Software* din *Bulgaria*.

Mircea Pașoi
319 puncte
argint



Adrian Vladu
170 de puncte
bronz

La concurs au participat 45 de concurenți din nouă țări: *Albania*, *Bulgaria*, *Cipru*, *Grecia*, *Macedonia*, *Republica Moldova*, *România*, *Serbia și Muntenegru* și *Turcia*.

Pe parcursul celor două zile de concurs, concurenții s-au confruntat cu câte trei probleme, pe parcursul a cinci ore.

Mai multe informații sunt disponibile pe *site*-ul oficial al concursului: **www.boi2004-plovdiv.org**.

Echipa României

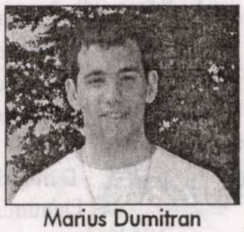
La *BOI* 2004, *România* a fost reprezentată de *Adrian Vladu*, *Mircea Pașoi*, *Silviu Gănceanu* și *Sorin-Marius Dumitran*.

Ei au reușit să obțină o medalie de argint și două de bronz.

Câștigătorii

Concursul a fost câștigat de bulgarul *Rangel Dokov* care a reușit să obțină 470 de puncte din 600 posibile.

Pe locurile următoare s-au clasat conaționalii săi *Valentin Mihov* și *Vladimir Nedev* cu 390, respectiv 366 de puncte.



Marius Dumitran
110 puncte
mențiune

Competiția a fost dominată de concurenții bulgari care au obținut patru medalii de aur, patru de argint și două de bronz.

Trebuie menționat faptul că țara gazdă a avut trei echipe în concurs (două reprezentative naționale și una a orașului *Plovdiv*).

Silviu Gănceanu
151 de puncte
bronz

Pe cea de-a doua poziție a clasamentului neoficial pe medalii s-a clasat echipa *Moldovei*, cu o medalie de aur, una de argint și două de bronz. A fost pentru prima dată la o competiție de acest nivel când echipa *Moldovei* s-a clasat în fața echipei noastre.

Echipa *României* a ocupat cea de-a patra poziție în acest clasament, ea fiind devansată de echipa *Turciei* care a reușit să obțină două medalii de argint și două de bronz.

În total, au fost acordate cinci medalii de aur, opt de argint și douăsprezece de bronz.

Pentru a obține medalia de aur au fost necesare 323 de puncte, pentru o medalie de argint au fost suficiente 246 de puncte, iar pentru una de bronz 127 puncte.

Configurații

La fel ca la majoritatea competițiilor internaționale de acest tip, la *BOI* 2004 compilatoarele utilizate au fost *FreePascal* și *GNU C/C++*.

Soluțiile concurenților au fost evaluate folosindu-se calculatoare dotate cu procesoare *Intel Celeron* având frecvența de 1,7 GHz.

Atena, septembrie 2004

Olimpiada INTERNAȚIONALĂ de Informatică 2004

Vă prezentăm în continuare enunțurile celor șase probleme propuse spre rezolvare la cea de-a șaisprezecea ediție a celui mai important concurs internațional de informatică destinat elevilor.

P040601: Artemis

Zeus i-a dăruit lui *Artemis*, zeița vânătorii, o zonă dreptunghiulară pentru a planta o pădure.

Zona este delimitată la stânga de un segment aflat pe semiaxa verticală pozitivă, iar la dreapta de un segment aflat pe semiaxa orizontală pozitivă.

Colțul din stânga-jos a hărții se află în punctul de coordonate (0, 0). *Zeus* i-a cerut zeiței să planteze copaci doar în puncte care au coordonate întregi.

Artemis a dorit ca pădurea să pară cât mai naturală, motiv pentru care a plantat pădurea astfel încât oricare linie care unește doi copaci să nu fie paralelă cu axele de coordonate.

Uneori *Zeus* dorește ca *Artemis* să taie copaci pentru el. Copacii trebuie tăiați respectându-se următoarele reguli:

- *Zeus* dorește să fie tăiați cel puțin *T* copaci.
- Pentru a obține o zonă în care se poate construi un teren de fotbal (*Zeus* dorește ca în viitor să obțină performanțe deosebite în acest domeniu), *Artemis* trebuie să taie toți copacii dintr-o zonă dreptunghiulară și să nu taie nici un copac din exteriorul zonei respective.
- Laturile zonei dreptunghiulare trebuie să fie paralele cu axele de coordonate.
- În două colțuri opuse ale zonei trebuie să se afle copaci și copacii din aceste colțuri trebuie și ei tăiați.

Datorită faptului că lui *Artemis* îi plac foarte mult copacii, ea dorește să îndeplinească aceste condiții tăind cât mai puțini copaci.

Va trebui să scrieți un program care, pe baza informațiilor disponibile despre pădure și a numărului minim *T* al copacilor care trebuie tăiați, determină zona dreptunghiulară din care *Artemis* va tăia copacii.

Date de intrare

Denumirea fișierului de intrare este **`artemis.in`**.

Prima linia a acestuia conține un număr întreg *N*, care reprezintă numărul copacilor din pădure.

Cea de-a doua linie conține un număr întreg *T* care reprezintă numărul minim al copacilor care trebuie tăiați.

Următoatele *N* linii descriu pozițiile celor *N* copaci. Fiecare dintre aceste linii conține două numere întregi *X* și *Y* care reprezintă coordonata orizontală, respectiv coordonata verticală, a unuia dintre copacii din pădurea plantată de către *Artemis*.

Date de ieșire

Denumirea fișierului de ieșire este **`artemis.out`**.

Acest fișier trebuie să conțină o singură linie pe care se vor afla două numere întregi *I* și *J*, separate printr-un spațiu.

Artemis ar trebui să folosească cel de-al *I*-lea copac (cel a cărui poziție este descrisă pe cea de-a (*I*+2)-a linie a fișierului de intrare) și cel de-al *J*-lea copac (cel a cărui poziție este descrisă pe cea de-a (*J*+2)-a linie a fișierului de intrare) pentru a identifica cele două colțuri opuse ale zonei dreptunghiulare din care vor fi tăiați copaci și unde va fi construit terenul de fotbal.

Ordinea celor două numere nu este importantă.

Dacă există mai multe soluții, trebuie determinată oricare dintre ele.

Se garantează existența cel puțin a unei soluții.

Restricții

Numărul copacilor din pădure este cuprins între 2 și 20.000.

Coordonatele copacilor din pădure sunt numere întregi cuprinse între 0 și 64.000.

Numărul copacilor care trebuie tăiați va fi întotdeauna cel mult egal cu numărul copacilor din pădurea plantată de *Artemis*.

Exemplu

`artemis.in`

```
3
2
1 1
2 3
5 6
```

`artemis.out`

```
1 2
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 16 MB

P040602: Hermes

Într-un oraș modern al zeilor eleni, străzile sunt dispuse sub forma unui caroiaj cu coordonate întregi, fiecare stradă fiind paralelă cu una dintre axele de coordonate.

Pentru fiecare număr întreg Z, există o stradă orizontală la $y = Z$ și o stradă verticală la $x = Z$.

Ca urmare, orice pereche de numere întregi reprezintă o intersecție de străzi.

În timpul zilelor caniculare, zeii se odihnesc la cafenelele din intersecții.

Curierul *Hermes* trebuie să trimită mesaje fotonice zeilor din cafenele deplasându-se numai de-a lungul străzilor.

Fiecare mesaj este destinat unui anumit zeu, dar nu contează dacă și alți zei primesc mesajul în momentul în care acesta este transmis de către *Hermes*.

Mesajele trebuie transmise într-o anumită ordine și *Hermes* cunoaște coordonatele cafenelelor în ordinea respectivă. *Hermes* pornește din punctul de coordonate (0, 0).

Pentru a trimite un mesaj la o cafenea aflată în punctul de coordonate (X_i, Y_i), este suficient ca *Hermes* să ajungă într-un punct al străzii orizontale pe care se află cafeneaua (cea a cărei coordonată verticală este Y_i) sau într-un punct al străzii verticale pe care se află cafeneaua (cea a cărei coordonată orizontală este X_i). După trimiterea tuturor mesajelor, *Hermes* se oprește.

Trebuie să scrieți un program care, dându-se o secvență de cafenele, determină distanța minimă pe care trebuie să o parcurgă *Hermes* pentru a trimite toate mesajele.

Date de intrare

Denumirea fișierului de intrare este `hermes.in`.

Prima linia a acestuia conține un număr întreg N, care reprezintă numărul mesajelor care trebuie trimise.

Următoatele N linii descriu pozițiile celor N cafenele la care trebuie trimise mesajele, în ordinea în care trebuie trimise aceste mesaje. Fiecare dintre aceste linii conține două numere întregi X și Y care reprezintă coordonata orizontală, respectiv coordonata verticală, a unei cafenele.

Restricții

Numărul mesajelor care trebuie trimise este cuprins între 1 și 20.000.

Coordonatele cafenelelor sunt numere cuprinse între -1000 și 1000.

Date de ieșire

Denumirea fișierului de ieșire este `hermes.out`.

Acest fișier trebuie să conțină o singură linie pe care se va afla un singur număr, reprezentând distanța minimă pe care trebuie să o parcurgă *Hermes*.

Exemplu

`hermes.in`

```
5
8 3
7 -7
8 1
-2 1
-2 1
6 -5
```

`hermes.out`

```
11
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 16 MB

P040603: Poligon

Un poligon este format din toate punctele aflate în interiorul său sau pe laturile sale.

Un poligon convex are proprietatea că, oricum am alege două dintre punctele sale, toate punctele care se află pe segmentul care unește cele două puncte sunt puncte ale poligonului.

Pentru această problemă vom lua în considerare doar poligoane convexe cu cel puțin două vârfuri, iar coordonatele vârfurilor sunt numere întregi.

Impunem și o condiție suplimentară: oricare trei vărfuri ale poligonului nu sunt coliniare.

Dându-se două poligoane A și B, suma *Minkovski* a celor două poligoane este un poligon format din toate punctele de forma $(x_1 + x_2, y_1 + y_2)$, unde (x_1, y_1) este un punct care face parte din poligonul A, iar (x_2, y_2) este un punct care face parte din poligonul B.

Se poate demonstra că suma *Minkovski* a două poligoane este întotdeauna un poligon.

În figura 1 este prezentat un exemplu: două triunghiuri și suma *Minkovski* a acestora.

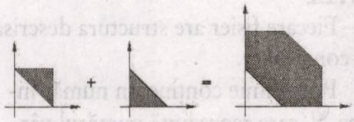


Figura 1

Ne interesează operația inversă sumei *Minkowski*. Pentru un poligon dat P, vom căuta două poligoane A și B astfel încât:

- Poligonul P reprezintă suma *Minkovski* a poligoanelor A și B.
- Poligonul A conține între două și patru vârfuri diferite. Cu alte cuvinte, este un segment (două vârfuri), un triunghi (trei vârfuri) sau un patrulater convex (patru vârfuri).
- Poligonul A trebuie să conțină cât mai multe vârfuri. Așadar, el trebuie să fie un patrulater dacă este posibil. Dacă nu poate fi patrulater, atunci trebuie să fie un triunghi, iar dacă nu poate fi nici triunghi, atunci trebuie să fie un segment (poligon degenerat cu două vârfuri).

Evident nici unul dintre poligoanele A și B nu poate fi identic cu P, deoarece celălalt poligon ar fi dat de un singur punct, ceea ce nu este permis.

Pentru această problemă sunt date mai multe fișiere de intrare, fiecare conținând o descriere a unui poligon P.

Pentru fiecare fișier de intrare trebuie determinate poligoanele A și B care respectă condițiile amintite anterior și trebuie creat un fișier de ieșire care conține descrierile poligoanelor A și B.

Se garantează existența poligoanelor A și B pentru toate poligoanele P descrise în fișierul de intrare.

Dacă există mai multe soluții posibile, poate fi aleasă oricare dintre ele.

Pentru această problemă nu trebuie scris nici un program, fiind suficientă generarea fișierelor de ieșire corecte.

Date de intrare

Sunt date zece fișiere de intrare, denumite **`poligon1.in`**, **`poligon2.in`** și așa mai departe, până la **`poligon10.in`**.

Fiecare fișier are structura descrisă în continuare.

Prima linie conține un număr întreg N, care reprezintă numărul vârfurilor poligonului P.

Următoarele N linii descriu vârfurile poligonului P, în ordinea inversă acelor de ceasornic. O astfel de linie descrie un vârf și conține două numere întregi X și Y care reprezintă coordonatele vârfului respectiv. Toate coordonatele sunt numere întregi nenegative.

Date de ieșire

Vor trebui generate zece fișiere de ieșire, denumite **`poligon1.out`**, **`poligon2.out`** și așa mai departe, până la **`poligon10.out`**.

Fiecare astfel de fișier va conține descrierea a două poligoane A și B, a căror sumă *Minkovski* este poligonul P din fișierul de intrare corespunzător.

Prima linie a unui astfel de fișier va trebui să conțină textul `#FILE polygon I`, unde I este un număr întreg cuprins între 1 și 10 și identifică numărul de ordine al fișierului respectiv. În continuare, structura fișierelor de ieșire este similară fișierelor de intrare.

Cea de-a doua linie trebuie să conțină un număr NA cuprins între 2 și 4, care reprezintă numărul vârfurilor poligonului A.

Următoarele NA linii trebuie să descrie vârfurile poligonului A, în ordinea inversă acelor de ceasornic. O astfel de linie va descrie un vârf și va conține două numere întregi X și Y care reprezintă coordonatele vârfului respectiv.

Linia următoare trebuie să conțină un număr NB mai mare sau egal cu 2, care reprezintă numărul vârfurilor poligonului B.

Următoarele NB linii trebuie să descrie vârfurile poligonului B, în ordinea inversă acelor de ceasornic. O astfel de linie va descrie un vârf și va conține două numere întregi X și Y care reprezintă coordonatele vârfului respectiv.

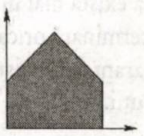
Exemplu

Vom considera că exemplu care urmează descrie fișierul cu numărul de ordine 0.

Datele de intrare sunt următoarele:

`polygon0.in`

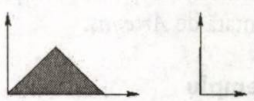
```
5
0 1
0 0
2 0
2 1
1 2
```



Pentru acest exemplu, ambele fișiere de ieșire prezentate în cele ce urmează sunt corecte. O primă soluție este:

`polygon0.out`

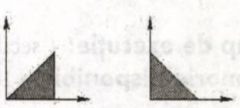
```
3
0 0
2 0
1 1
2
0 1
0 0
```



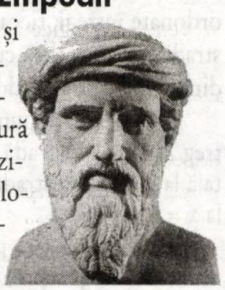
Cea de-a doua soluție este următoarea:

`polygon0.out`

```
3
0 0
1 0
1 1
3
0 1
0 0
1 0
```



P040604: Empodii



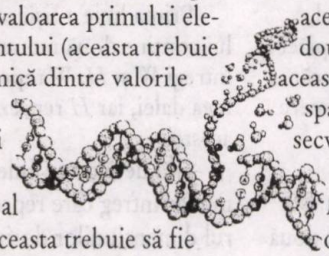
Matematicianul și filosoful *Pitagora* credea că realitatea are o natură matematică. În zilele noastre, biologii studiază proprietățile secvențelor biologice.

O secvență biologică este o succesiune de M numere întregi, astfel încât:

- Secvența conține toate numerele cuprinse între 0 și M - 1.
- Secvența începe cu 0 și se termină cu M - 1.
- Nu există două elemente de forma E și E + 1 aflate pe poziții consecutive în această ordine.

O subsecvență formată din elemente adiacente ale unei secvențe biologice este numită ***segment***.

Un segment al unei secvențe biologice este numit ***interval mărginit*** dacă include toate numerele întregi cuprinse între valoarea primului element al segmentului (aceasta trebuie să fie cea mai mică dintre valorile elementelor segmentului) și valoarea ultimului element al segmentului (aceasta trebuie să fie cea mai mare dintre valorile elementelor segmentului).



Un interval mărginit este numit ***empodiu*** dacă în cadrul său nu apare nici un alt interval mărginit mai scurt.

De exemplu, să considerăm secvența biologică (0, 3, 5, 4, 6, 2, 1, 7). Se observă că întreaga secvență biologică este un interval mărginit. Totuși, ea conține un alt interval mărginit, și anume (3, 5, 4,6), deci nu este un empodiu. Intervalul mărginit (3, 5, 4, 6) nu conține nici un alt interval mărginit, deci este un empodiu.

Mai mult, acesta este singurul empodiu din cadrul secvenței biologice considerate.

Trebuie să scrieți un program care, dându-se o secvență biologică, determină toate empodiile din cadrul secvenței biologice respective.

Date de intrare

Denumirea fișierului de intrare este **`empodia.in`**.

Prima linie a fișierului conține un singur număr întreg M care reprezintă numărul elementelor secvenței biologice.

Următoarele M linii conțin numerele întregi ale secvenței biologice, în ordinea în care apar acestea în secvență. Fiecare dintre aceste linii conține un singur număr întreg.

Date de ieșire

Denumirea fișierului de ieșire este **`empodia.out`**.

Prima linie a acestui fișier trebuie să conțină un număr întreg H care reprezintă numărul empodiilor din secvența biologică.

Următoarele H linii descriu toate empodiile din secvența biologică dată, în ordinea apariției acestora în secvența biologică. Fiecare dintre aceste linii trebuie să conțină două numere întregi A și B (în această ordine), separate printr-un spațiu, unde al A-lea element al secvenței biologice este primul element al empodiului, iar al B-lea elemenet al secvenței biologice este ultimul element al empodiului.

Restricții

Numărul elementelor din secvența biologică este cuprins între 2 și 1.100.000.

Exemplu

`empodia.in`

```
8
0
3
5
4
6
2
1
7
```

`empodia.out`

```
1
2 5
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 128 MB

P040605: Fermierul

Un fermier are mai multe terenuri, fiecare fiind înconjurat de chiparoși. De asemenea, fermierul are mai multe fâșii de pământ, fiecare conținând o linie de chiparoși.

Atât pe terenuri cât și pe fâșii, între oricare doi chiparoși consecutivi (învecinați) se află câte un măslin.

Toți chiparoșii fermierului înconjoară un teren sau se află pe o fâșie și toți măslinii fermierului se află între doi chiparoși consecutivi de pe un teren sau de pe o fâșie.

Într-o zi fermierul s-a îmbolnăvit și a simțit că i se apropie sfârșitul.

Când mai avea doar câteva zile de trăit l-a chemat pe fiul cel mare și i-a spus:

"Îți voi da oricare Q chiparoși pe care îi dorești și toți măslinii care se află între doi chiparoși consecutivi dintre cei aleși."

De pe fiecare teren și de pe fiecare fâșie, fiul poate alege orice combinație de chiparoși.

Datorită faptului că fiului îi plac foarte mult măslinele, el dorește să aleagă cei Q chiparoși astfel încât să primească un număr cât mai mare de măslini.

Pentru exemplul din figura 2 considerăm că fiul primește 17 chiparoși. Pentru a maximiza numărul de măslini pe care îi va primi, fiul va trebui să aleagă toți chiparoșii de pe primele două terenuri. Astfel, fiul va reuși să obțină 17 măslini.

Trebuie să scrieți un program care, pe baza informațiilor referitoare la terenurile și la fâșiile fermierului și a numărului de chiparoși pe care îi poate alege fiul, determină cel mai mare număr de măslini pe care îi poate primi fiul.

Date de intrare

Denumirea fișierului de intrare este **`farmer.in`**.

Prima linie a fișierului conține trei numere întregi Q, M și K. Q reprezintă numărul chiparoșilor pe care îi poate alege fiul, M reprezintă numă-

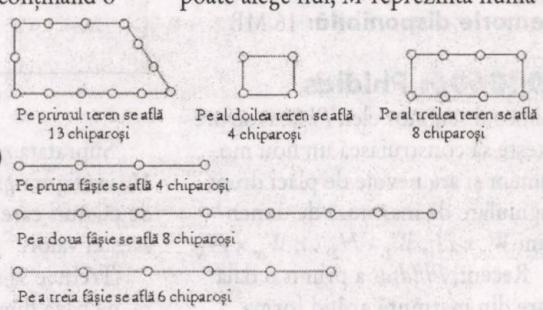


Figura 2

rul terenurilor, iar K reprezintă numărul fâșiilor.

Cea de-a doua linie conține M numere întregi, reprezentând numărul de chiparoși de pe fiecare dintre cele M terenuri.

Cea de-a treia linie conține K numere întregi, reprezentând numărul de chiparoși de pe fiecare dintre cele K fâșii.

Date de ieșire

Denumirea fișierului de ieșire este **farmer.out**.

Fișierul va conține o singură linie pe care se va afla un singur număr reprezentând numărul maxim al măslinilor pe care îi poate primi fiul fermierului.

Restricții și precizări

Numărul chiparoșilor care trebuie aleși este cuprins între 0 și 150.000.

Numărul terenurilor și numărul fâșiilor sunt cuprinse între 0 și 2000.

Pe fiecare teren se află între 3 și 150 de chiparoși.

Pe fiecare fâșie se află între 2 și 150 de chiparoși.

Exemplu

farmer.in

```
17 3 3
13 4 8
4 8 6
```

farmer.out

```
17
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 16 MB

P040606: Phidias

Faimosul sculptor elen *Phidias* se pregătește să construiască un nou monument și are nevoie de plăci dreptunghiulare de marmură de dimensiuni $W_1 \times H_1$, $W_2 \times H_2$, ..., $W_N \times H_N$.

Recent, *Phidias* a primit o dală mare din marmură având forma dreptunghiulară. El vrea să taie dala astfel încât să obțină plăcile de dimensiunile dorite.

Fiecare bucată de marmură (dala sau bucățile tăiate din aceasta) poate fi tăiată fie orizontal, fie vertical, obținându-se două plăci dreptunghiulare ale căror dimensiuni sunt numere întregi.

Fiecare tăietură trebuie să fie completă. Aceasta este singura modalitate de tăiere, iar două plăci tăiate nu pot fi lipite ulterior pentru a obține o nouă placă.

Datorită faptului că pe dala de marmură primită este desenat un model, plăcile tăiate nu pot fi rotite.

Dacă *Phidias* taie o placă de dimensiuni $A \times B$, ea nu poate fi folosită în locul unei plăci de dimensiuni $B \times A$ decât dacă $A = B$.

Pentru fiecare dimensiune specificată, *Phidias* poate tăia nici una, una, două sau mai multe plăci care au dimensiunea respectivă.

O placă este pierdută dacă nu are nici una dintre dimensiunile dorite după ce s-au efectuat toate tăieturile.

Phidias se întreabă cum poate tăia dala astfel încât să se piardă cât mai puțină marmură.

De exemplu, să presupunem că în figura 3 lungimea dalei este 21, iar lățimea este 11.

Phidias dorește să construiască dale de dimensiuni 10×4, 6×2, 7×5 și 15×10.

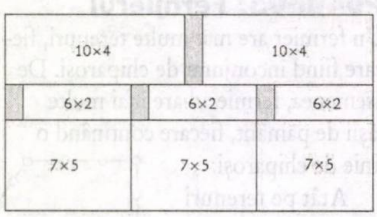


Figura 3

Suprafața minimă pierdută este 10, iar figura prezintă o succesiune de tăieturi care duce la obținerea acestei valori.

Trebuie să scrieți un program care, pe baza dimensiunilor dalei și ale pieselor care trebuie tăiate, determină suprafața minimă care este pierdută.

Date de intrare

Denumirea fișierului de intrare este **phidias.in**.

Prima linie a fișierului conține două numere întregi W și H. W reprezintă lungimea dalei, iar H reprezintă lățimea acesteia.

Cea de-a doua linie conține un număr întreg care reprezintă numărul dimensiunilor dorite pentru plăcile tăiate.

Următoarele N linii conțin dimensiunile dorite ale plăcilor. Fiecare dintre aceste linii conține câte două numere întregi. Primul dintre ele reprezintă lungimea dorită a piesei, iar al doilea reprezintă lățimea dorită.

Date de ieșire

Denumirea fișierului de ieșire este **phidias.out**.

Fișierul va conține o singură linie pe care se va afla un singur număr reprezentând suprafața minimă care este pierdută.

Restricții

Dimensiunile dalei sunt numere întregi cuprinse între 1 și 600.

Numărul dimensiunilor dorite este cuprins între 1 și 200.

Lățimile plăcilor care trebuie obținute sunt cuprinse între 1 și lățimea dalei, în timp ce lungimile acestor plăci sunt cuprinse între 1 și lungimea dalei.

Exemple

phidias.in

```
21 11
4
10 4
6 2
7 5
15 10
```

phidias.out

```
10
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 16 MB

Rzeszów, iulie 2004

OLIMPIADA de Informatică a EUROPEI CENTRALE 2004

Vă prezentăm în continuare enunțurile celor șase probleme propuse spre rezolvare la cea de-a unsprezecea ediție a acestui concurs care este considerat, pe bună dreptate, ca fiind una dintre cele mai puternice competiții destinate elevilor informaticieni.

P040607: Nori

Pe cer sunt n nori care se deplasează în aceeași direcție, cu o viteză constantă $v = (v_x, v_y)$. Așadar, pentru orice număr real pozitiv t și orice punct al unui nor ale cărui coordonate inițiale sunt (x, y), poziția punctului respectiv la momentul t este $(x + t \cdot v_x, y + t \cdot v_y)$.

Pentru simplitate, vom presupune că fiecare nor este reprezentat de un poligon (norul este dat de interiorul și de frontiera poligonului), ale cărui vârfuri au coordonatele întregi.

Poligoanele nu trebuie să fie neapărat convexe, dar oricare două laturi nu se intersectează decât, eventual, extremitățile lor în vârfurile poligonului. Poligoanele care descriu norii pot avea puncte comune.

Pe pământ (în punctul de origine al axelor de coordonate) se află un centru de control al sateliților și exact deasupra centrului (și a norilor) se află un satelit.

De la centrul de control este trimisă o rază laser verticală pentru a se realiza comunicarea cu satelitul. În cazul în care raza laser intersectează un nor, comunicarea cu satelitul este întreruptă.

Se știe că la momentul 0, raza laser, nu intersectează nici un nor. În timpul transmisiilor, pot exista mai multe momente când raza laser intersectează norii (unul sau mai mulți) întrerupându-se comunicarea cu satelitul.

Comunicarea este întreruptă chiar dacă raza laser intersectează un vârf sau o latură a unui poligon care reprezintă un nod.

Trebuie să scrieți un program care determină numărul total al întreruperilor care au loc până în momentul în care norii părăsesc zona de observație.

Date de intrare

Datele se citesc de la intrarea standard.

Pe prima linie se vor afla trei numere întregi n, v_x și v_y, separate prin câte un spațiu, reprezentând numărul norilor, viteza norilor pe axa orizontală, respectiv viteza norilor pe axa verticală.

Următoarele n linii conțin date despre nori. Fiecare dintre aceste linii descrie câte un nor și conține un șir de numere întregi, separate prin câte un singur spațiu. Primul număr întreg k, de pe o astfel de linie reprezintă numărul vârfurilor care reprezintă norul corespunzător liniei. Urmează k perechi de numere întregi $(x_1, y_1, x_2, y_2, \ldots, x_k, y_k)$ care reprezintă coordonatele celor k vârfuri ale poligonului. Aceste vârfuri sunt descrise în ordinea dată de sensul arcelor de ceasornic.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard. Va fi scrisă o singură linie care va conține un singur număr, reprezentând numărul întreruperilor de comunicare.

Restricții

Numărul norilor este cuprins între 1 și 1000.

Valorile vitezelor pe cele două axe de coordonate sunt numere întregi cuprinse între -1.000.000.000 și 1.000.000.000 și nu pot fi ambele nule.

Numărul vârfurilor unui poligon care descrie un nor este cuprins între 3 și 1000.

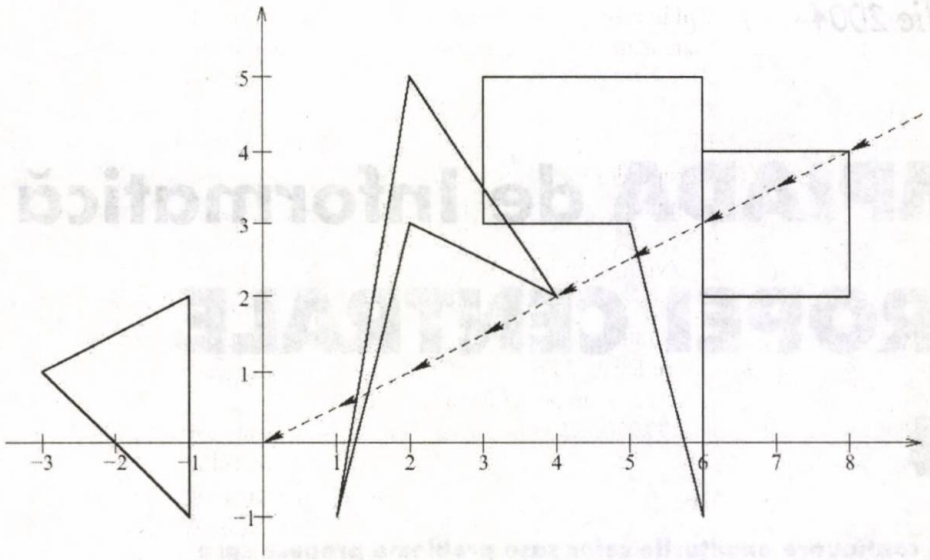


Figura 1

Coordonatele punctelor care reprezintă vârfurile poligoanelor corespunzătoare norilor sunt numere întregi cuprinse între -1.000.000.000 și 1.000.000.000.

Abscisele corespund direcției *Vest - Est*, iar ordonatele corespund direcției *Nord - Sud*.

Exemplu

Intrarea standard

```
4 -2 -1
4 6 2 6 4 8 4 8 2
4 2 3 1 -1 2 5 4 2
3 -3 1 -1 -2 -1 -1
5 5 3 3 3 3 5 6 5 6 -1
```

Ieșirea standard

```
3
```

Configurația din acest exemplu este prezentată în figura 1.

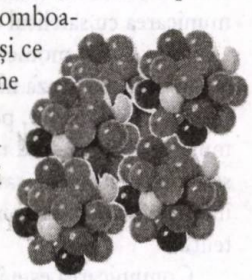
Timp de execuție: 3 secunde/test
Memorie disponibilă: 64 MB

P040608: Bomboane

John are *n* bomboniere în care își ține bomboanele. Fiecare dintre bomboniere conține un anumit tip de bomboane, dar nici o altă bomboniera nu conține bomboane de acel tip). Se cunoaște numărul de bomboane din fiecare bomboniera.

John s-a hotărât să mănânce o parte dintre bomboanele sale. El ar vrea să mănânce cel puțin *a* bomboane, dar nu mai mult decât *b* bomboane.

Problema este că *John* nu se poate hotărî câte bomboane să mănânce și ce tip de bomboane să aleagă.



În câte moduri poate el să facă acest lucru?

Date de intrare

Datele se citesc de la intrarea standard.

Pe prima linie se vor afla trei numere întregi *n*, *a* și *b*, separate prin câte un spațiu, reprezentând numărul bombonierelor, numărul minim de bomboane care trebuie alese, respectiv numărul maxim de bomboane care trebuie alese.

Fiecare din următoarele *n* linii conține un număr întreg care reprezintă numărul de bomboane dintr-o anumită bomboniera.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Va fi scrisă o singură linie care va conține un singur număr, reprezentând numărul posibilităților pe care le are *John* pentru a alege bomboanele.

Restricții

Numărul bomboanierelor este cuprins între 1 și 10.

Numărul minim, precum și numărul maxim de bomboane pot varia între 0 și 10.000.000.

În fiecare bomboniera se pot afla între 0 și 1.000.000 de bomboane.

Exemplu

Intrarea standard

```
2 1 3
3
5
```

Ieșirea standard

```
9
```

Explicație

Cele nouă posibilități de a alege sunt:

Timp de execuție: 0.1 secunde/test
Memorie disponibilă: 16 MB

P040609: Excursii

În vacanța care urmează, o mulțime de oameni vor să plece în excursii de neuitat.

Pentru ca vacanța să fie cât mai plăcută, fiecare vrea să meargă însoțit de un grup de prieteni.

O agenție de turism oferă diferite excursii. Fiecare excursie se adresează unui grup de persoane, care tre-

buie să se încadreze între un număr minim și un număr maxim de participanti.

Fiecare grup poate să aleagă o singură excursie. În plus, fiecare excursie poate fi aleasă de un singur grup.

Agenția de turism dorește să organizeze cât mai multe excursii.

Sarcina voastră este să alegeți câte o excursie pentru fiecare grup, astfel încât să se organizeze un număr cât mai mare de excursii.

Date de intrare

Datele se citesc de la intrarea standard.

Prima linie conține două numere întregi n și m separate printr-un spațiu, unde n este numărul de grupuri și m este numărul de excursii. Grupurile sunt numerotate de la 1 la n, iar excursiile de la 1 la m.

Următoarele n linii conțin dimensiunile grupurilor. Fiecare linie conține un singur număr întreg, iar ordinea liniilor respectă numerele de ordine ale grupurilor.

Următoarele m linii conțin descrierile excursiilor. Fiecare linie conține două numere întregi l și u, separate printr-un singur spațiu, unde l reprezintă dimensiunea minimă a unui grup care poate merge în excursia respectivă, iar u reprezintă dimensiunea maximă a unui astfel de grup. Ordinea liniilor respectă numerele de ordine ale excursiilor.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Prima linie va conține un număr întreg k care reprezintă numărul maxim al excursiilor care pot fi organizate.

Fiecare dintre următoarele k linii vor conține câte o pereche de numere întregi, separate printr-un singur spațiu. Primul dintre ele va fi numărul de ordine al unui grup, iar al doilea va fi numărul de ordine al unei excursii. Membrii grupului specificat vor alege excursia aleasă.

În cazul în care există mai multe soluții, poate fi aleasă oricare dintre ele.

Restricții

Numărul grupurilor și numărul excursiilor sunt cuprinse între 1 și 400.000.

Numărul minim, precum și numărul maxim de membri ai unui grup pot varia între 1 și 1.000.000.000.

Din fiecare grup pot face parte între 1 și 1.000.000.000 de persoane.

Exemplu

Intrarea standard

```
5 4
54
6
9
42
15
6 6
20 50
2 8
7 20
```

Ieșirea standard

```
3
2 1
3 4
4 2
```

Timp de execuție: 3 secunde/test
Memorie disponibilă: 16 MB

P040610: Fotbal

Într-un campionat de fotbal sunt înscrise un număr de n echipe (n este număr par). Echipele sunt identificate prin numere întregi cuprinse între 1 și n.

Pe parcursul unui sezon fiecare echipă joacă cu fiecare dintre celelalte echipe, exact o dată. Așadar, sezonul constă din n - 1 etape.

Este de dorit ca fiecare echipă să joace meciurile consecutive pe stadioane diferite, adică un meci pe stadionul propriu, meciul următor în deplasare, apoi din nou pe stadionul propriu și așa mai departe.

Din nefericire nu este posibilă o planificare, astfel încât să nu existe echipe care joacă două meciuri consecutive acasă sau două meciuri consecutive în deplasare. Când se face o planificare a meciurilor dintr-un sezon, numărul acestor situații trebuie minimizat.

De exemplu, dacă o echipă joacă în deplasare, apoi de patru ori acasă, apoi în deplasare, se consideră că a jucat de trei ori două meciuri consecutive numai acasă sau numai în deplasare).

Programul vostru trebuie să determine numărul minim al situațiilor în care o echipă joacă două meciuri consecutive acasă sau două meciuri consecutive în deplasare și să construiască o planificare a meciurilor sezonului astfel încât să fie respectat acest număr minim.

Sezonul constă în n - 1 etape. O etapă constă în n / 2 partide, fiecare echipă jucând exact un meci într-o etapă.

Pe parcursul întregului sezon se vor desfășura $n \cdot (n - 1) / 2$ partide. Fiecare dintre acestea se joacă pe unul dintre stadioanele celor două echipe participante (nu se joacă pe stadioane neutre).

Date de intrare

Datele se citesc de la intrarea standard.

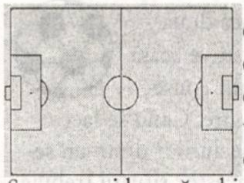
Intrarea constă într-o singură linie pe care se va afla un singur număr, reprezentând numărul echipelor participante.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Pe prima linie va fi scris un singur număr întreg care va reprezenta numărul minim al situațiilor în care o echipă joacă două meciuri consecutive acasă sau două meciuri consecutive în deplasare.

Următoarele n - 1 linii vor conține o planificare a jocurilor. Fiecare linie va corespunde unei etape, iar liniile vor fi scrise în ordinea desfășurării etapelor. O astfel de linie va conține toate cele n numere de



ordine ale echipelor (fiecare număr va apărea exact o dată). Se va considera că echipele aflate pe o poziție impară în cadrul liniei vor juca acasă cu echipa de pe poziția imediat următoare.

Restricție

Numărul echipelor care participă în campionat este cuprins între 2 și 1000.

Exemplu

Intrarea standard

```
4
```

Ieșirea standard

```
2
1 2 3 4
4 1 2 3
1 3 4 2
```

Timp de execuție: 3 secunde/test
Memorie disponibilă: 32 MB

P040611: Puzzle

Regele din *Byteland* a primit cadou un *puzzle*. Acesta are forma unui pătrat de latură n și conține n^2 celule pătratice, dispuse pe n linii și n coloane, fiecare celulă având latura egală cu unitatea.

Fiecare celulă conține o piesă cu un număr cuprins între 1 și n^2. Numerele pieselor din fiecare celulă a *puzzle*-ului sunt distincte.

Liniile și coloanele *puzzle*-ului sunt numerotate de la 1 la n; fiecare celulă va fi identificată printr-o pereche de numere întregi i și j, iar numărul din celula respectivă va fi notat prin $p(i, j)$.

O reprezentare schematică a *puzzle*-ului este prezentată în figura 2.

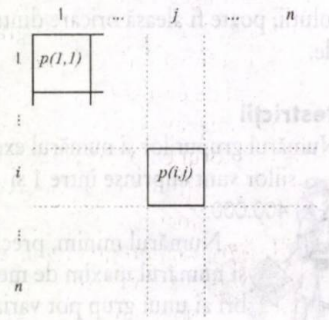


Figura 2

Pentru a rezolva *puzzle*-ul trebuie așezate piesele în ordine astfel încât pentru toate celulele *puzzle*-ului să avem:

$$p(i, j) = i \cdot n + j.$$

Mutările permise pentru rezolvarea *puzzle*-ului sunt:

- o permutare circulară a celulelor unei linii la dreapta cu un anumit număr de celule;
- O permutare circulară a celulelor unei coloane în jos cu un anumit număr de celule.

Regele din *Byteland* a reușit să rezolve *puzzle*-ul, dar el nu este sigur dacă l-ar putea rezolva pornind de la o configurație inițială diferită.

Ajutați regele să rezolve *puzzle*-ul pentru o configurație inițială specificată.

Date de intrare

Datele se citesc de la intrarea standard.

Pe prima linie se află un număr întreg n care reprezintă dimensiunea *puzzle*-ului.

Următoarele linii conțin descrierea configurației inițiale.

Fiecare dintre aceste linii conține câte n numere întregi, separate prin câte un spațiu, care reprezintă numerele de pe piesele de pe linia respectivă a configurației inițiale a *puzzle*-ului pe care l-a primit regele.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Dacă nu există nici o soluție, se va scrie o singură linie conținând numai cuvântul NO.

Dacă există cel puțin o soluție, prima linie va conține un număr întreg m care reprezintă numărul de mutări care conduc la rezolvarea *puzzle*-ului.

Următoarele m linii vor conține descrierea mutărilor, câte o mutare pe o linie. Fiecare astfel de linie conține o literă R (pentru permutarea circulară a unei linii la dreapta) sau litera C (pentru permutarea circulară a unei coloane în jos) un spațiu și două numere întregi k și l, unde k reprezintă numărul de ordine al liniei sau coloanei care va fi permutată, iar l reprezintă numărul de poziții cu care este permutată aceasta (liniile sunt permutate spre dreapta, iar coloanele sunt permutate în jos).

Restricții

Dimensiunea *puzzle*-ului este un număr întreg cuprins între 2 și 200.

Numărul maxim de mutări permise este 400.000 indiferent de configurație.

Exemplu

Intrarea standard

```
4
4 6 2 3
5 10 7 8
9 14 11 12
13 1 15 16
```

Ieșirea standard

```
2
C 2 1
R 1 3
```

Explicație

În continuare este prezentată configurația inițială a *puzzle*-ului, precum și configurațiile obținute după fiecare

dintre mutările care trebuie efectuate:

4	6	2	3
5	10	7	8
9	14	11	12
13	1	15	16

4	1	2	3
5	6	7	8
9	10	11	12
13	14	15	16

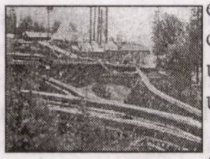
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Timp de execuție: 0.3 secunde/test
Memorie disponibilă: 32 MB

P040612: Cherestea

De-a lungul unei șosele, care pornește din vârful unui deal și ajunge la poalele acestuia au fost plantați n copaci.

La un moment dat, administrația locală a decis să taie copacii de la marginea șoselei. Pentru a nu irosi lemn, s-a decis că, după tăiere, copacii vor fi prelucrați în fabrici de cherestea.



Pentru ca efortul necesar transportului să fie cât mai mic, copacii vor fi transportați doar de sus în jos (dinspre vârf spre poalele dealului).

La poalele dealului se află o fabrică de prelucrat cheresteaua. Pentru a micșora distanțele pe care ar trebui transportat lemnul, administrația locală a decis să se construiască două alte fabrici de cherestea de-a lungul șoselei.

Va trebui să determinați pozițiile celor două fabrici de cherestea astfel încât costul necesar pentru a transporta lemnul să fie cât mai mic posibil.

Evident, fiecare copac va fi transportat la cea mai apropiată fabrică de cherestea a cărei poziție este mai jos decât poziția copacului respectiv.

Costul transportului este de un eurocent pentru a transporta un kilogram de lemn pe o distanță de un metru.

Date de intrare

Datele se citesc de la intrarea standard.

Prima linie a fișierului de intrare conține un număr întreg n care reprezintă numărul copacilor plantați de-a lungul șoselei.

Cei n copaci vor fi identificați prin numere întregi cuprinse între 1 și n, începând cu copacul aflat cel mai aproape de vârful dealului.

Fiecare dintre următoarele n linii conține câte două numere întregi pozitive, separate prin câte un spațiu. Cele două numere indică, în această ordine, greutatea copacului exprimată în kilograme, respectiv distanța, exprimată în metri, dintre copacul respectiv și copacul următor (cel descris pe linia următoare).

Pentru ultimul copac (cel descris pe ultima linie), distanța va fi cea dintre copacul respectiv și fabrica de cherestea aflată la poalele dealului.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Va fi scrisă o singură linie care va conține un singur număr, reprezentând costul total minim al transportului (exprimat în eurocenți) după ce vor fi construite cele două fabrici de cherestea.

Restricții

Numărul copacilor plantați la marginea șoselei este cuprins între 2 și 20.000.

Se garantează faptul că, în orice situație, costul total minim al transportului nu va depăși 2.000.000.000.

Exemple

Intrarea standard

```
9
1 2
2 1
3 3
1 1
3 2
1 6
2 1
1 2
1 1
```

Ieșirea standard

```
26
```

Explicație

În figura 3 sunt prezentate pozițiile optime ale fabricilor de cherestea pentru configurația specificată în exemplul anterior.

Copacii sunt reprezentați prin cercuri, iar greutățile lor sunt specificate sub cercurile respective.

Fabricile de cherestea sunt marcate cu negru și sunt poziționate în această figură în poziții în care se află copaci.

Rezultatul este:

$$1 \cdot (2 + 1) + 2 \cdot 1 + 1 \cdot (1 + 2) + 3 \cdot 2 + 2 \cdot (1 + 2 + 1) + 1 \cdot (2 + 1) + 1 \cdot 1 = 26$$

Timp de execuție: 1 secundă/test
Memorie disponibilă: 16 MB

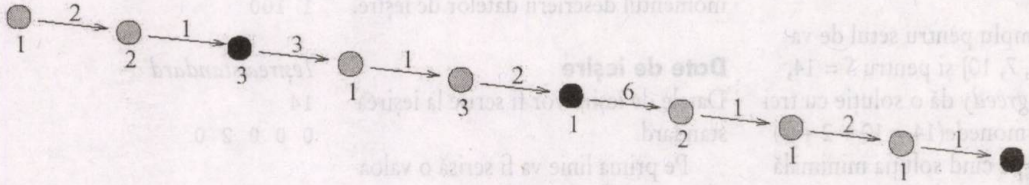


Figura 3

Plovdiv, iulie 2004

Olimpiada BALCANICĂ de Informatică 2004

Vă prezentăm în continuare enunțurile celor șase probleme propuse spre rezolvare la cea de-a douăsprezecea ediție a competiției la care se întâlnesc cei mai merituoși elevi informaticieni din Peninsula Balcanică.

P040613: Monede

Se consideră M numere întregi pozitive dintre care unul este 1. Există, de asemenea, un număr nelimitat de monede pentru fiecare din valorile date.

Se consideră următoarea problemă:

O anumită sumă S trebuie plătită folosind un număr minim de monede cu valorile date.

Este cunoscut faptul că această problemă poate fi rezolvată în anumite cazuri folosind următorul algoritm *greedy*:

Se caută moneda cu cea mai mare valoare care este mai mică sau egală cu suma S *după care se scade valoarea monedei găsite din* S. *Se continuă în aceeași manieră până când* S *devine* 0. *Numărul de monede folosit de algoritm pentru a reduce* S *la* 0 *pare să fie numărul minim de monede necesare.*

În multe cazuri algoritmul de mai sus este eficient, dar pentru anumite seturi de valori ale monedelor și anumite valori ale sumei S, algoritmul descris nu duce la obținerea soluției optime.

De exemplu pentru setul de valori {1, 2, 5, 7, 10} și pentru $S = 14$, algoritmul *greedy* dă o soluție cu trei monede(14 = 10 + 2 + 2) pe cînd soluția minimală este cu două monede (14 = 7 + 7).

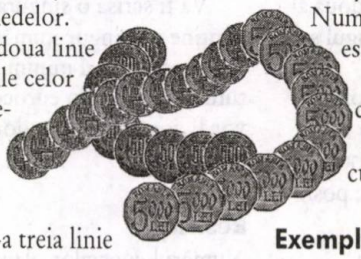
Astfel apare problema: *pentru ce set de valori ale monedelor algoritmul greedy nu produce o soluție corectă?*

Scrieți un program care, pentru un set dat de valori ale monedelor, decide dacă există o sumă S care poate fi obținută folosind un număr de monede mai mic decât cel obținut folosind algoritmul *greedy* prezentat anterior.

Date de intrare

Datele se citesc de la intrarea standard.

Prima linie conține numărul M al valorilor monedelor.



Cea de-a doua linie conține valorile celor M monede, separate prin câte un spațiu.

Pe cea de-a treia linie se află două numere întregi x și y, separate printr-un spațiu. Semnificația acestor valori va fi prezentată în momentul descrierii datelor de ieșire.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Pe prima linie va fi scrisă o valoare S, cuprinsă între valorile x și y specificate la intrare, pentru care algoritmul *greedy* nu duce la obținerea soluției optime.

Cea de-a doua linie trebuie să conțină numerele $b_1, b_2, \ldots, b_M$ care reprezintă numărul de monede de fiecare valoare (în aceeași ordine ca la citire) folosite pentru obținerea sumei S. Numărul total de monede folosite trebuie să fie mai mic decât numărul de monede obținut folosind algoritmul *greedy*.

În cazul în care există mai multe soluții trebuie generată doar una dintre ele.

Se garantează faptul că datele de intrare vor fi astfel alese încât pentru cel puțin una dintre valorile S cuprinse între x și y, algoritmul *greedy* nu va duce la obținerea soluției optime.

Restricții

Numărul tipurilor de monede este cuprins între 1 și 100.
Valorile monedelor sunt cuprinse între 1 și 7.000.000.
Valorile x și y sunt cuprinse între 1 și 7.000.000.

Exemplu

Intrarea standard

```
5
1 2 5 7 10
1 100
```

Ieșirea standard

```
14
0 0 0 2 0
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

P040614: Echipa

În scurt timp va avea loc *Olimpiada Interpeninsulară de Informatică*, iar liderii echipei *Peninsulei Balcanice* doresc să aleagă cei mai buni concurenți din peninsulă pentru a obține performanțe cât mai bune.

Din fericire echipa poate fi aleasă dintre N candidați, care sunt identificați prin numere întregi cuprinse între 1 și N.

Pentru a selecta cei mai buni candidați au fost organizate trei competiții.

Fiecare dintre cei N candidați a participat la toate competițiile, iar la fiecare competiție nu au existat doi sau mai mulți concurenți aflați la egalitate.

Spunem că un concurent A este ***mai bun*** decât un concurent B dacă A s-a clasat înaintea lui B în toate competițiile.

Un concurent A este excelent dacă nici un concurent nu este mai bun decât A.

Pentru a forma cea mai bună echipă, liderii echipei *Peninsulei Balcanice* doresc să afle numărul concurenților excelenți pe care îi au la dispoziție.

Va trebui să scrieți un program care, pe baza unei valori N date și rezultatele celor trei competiții, determină numărul concurenților excelenți.

Date de intrare

Datele se citesc de la intrarea standard.

Pe prima linie se află numărul M al concurenților din *Peninsula Balcanică*.

Fiecare dintre următoarele trei linii conține clasamentul uneia dintre cele trei competiții care au fost organizate.

O astfel de linie conține numerele de ordine ale concurenților, separate prin câte un spațiu, în ordinea în care s-au clasat aceștia în competiția corespunzătoare liniei.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Va fi scrisă o singură linie care va conține un singur număr, reprezentând numărul concurenților excelenți din *Pensinsula Balcanică*.

Restricție

Numărul concurenților este cuprins între 3 și 500.000.

Exemple

Intrarea standard

```
3
2 3 1
3 1 2
1 2 3
```

Ieșirea standard

```
3
```

Intrarea standard

```
10
2 5 3 8 10 7 1 6 9 4
1 2 3 4 5 6 7 8 9 10
3 8 7 10 5 4 1 2 6 9
```

Ieșirea standard

```
4
```

Explicații

Pentru primul exemplu toți concurenții sunt excelenți deoarece nu există nici un concurent care este mai bun decât altul (pe rând, fiecare concurent s-a aflat înaintea celorlalți doi în câte una dintre cele trei competiții).

Pentru cel de-al doilea exemplu, cei patru concurenți excelenți sunt cei identificați prin numerele 1, 2, 3 și 5.

Timp de execuție: 2.5 secunde/test

Memorie disponibilă: 64 MB

P040615: Drum minim

Compania *Trimoncium Soft* vrea să intre pe piața de *software* cu un produs pentru calculatoare de buzunar cu memorie limitată, care este capabil să presteze o serie de servicii utile via *Internet*.

Un astfel de serviciu oferă utilizatorului posibilitatea de a găsi cea mai scurtă cale între două intersecții ale unui oraș.

Din cauza dificultăților tehnice aplicația ar trebui să facă cât mai puține cereri prin *Internet*.

În oraș se află N intersecții, identificate prin numere întregi cuprinse între 1 și N.

Poziția unei intersecții C este dată de o pereche de coordonate întregi X_C și Y_C într-un sistem de coordonate ortogonal.

Pot exista două sau mai multe intersecții diferite cu aceleași coordonate. Dacă între două astfel de intersecții nu există o stradă, atunci nu se poate trece direct de la o intersecție la cealaltă.

De la fiecare intersecție C se poate ajunge la alte M_C intersecții, numite vecini ai intersecției C, prin străzi pe care se poate circula într-un singur sens. Fiecare stradă pe care se poate circula în ambele sensuri este descrisă de două străzi pe care se poate circula într-un singur sens (sensurile celor două străzi sunt opuse).

Este posibil ca două străzi să se intersecteze, dar punctul de intersecție să nu fie o intersecție de străzi. De exemplu, ele pot fi plasate pe diferite niveluri (există poduri sau tuneluri).

Lungimea unei străzi este distanța dintre cele două intersecții legate de această stradă.

Scrieți un program care, pentru două intersecții date S și F determină, folosind subprogramele unei biblioteci, cel mai scurt drum de la intersecția S până la intersecția F.

Numerele întregi N, S și F sunt obținute prin intermediul celor trei argumente ale subprogramului `start`. Acest subprogram trebuie apelat înaintea oricărui alt subprogram.

Coordonatele X_C și Y_C, precum și numărul M_C al vecinilor unei intersecții C pot fi obținute prin intermediul ultimelor trei argumente ale

subprogramului `getXYM`. Primul argument al acestui subprogram este valoarea *C*.

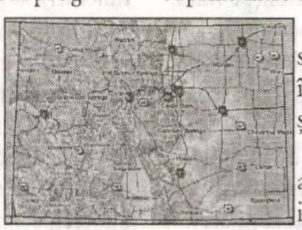
Vecinii unei intersecții *C* sunt obținuți prin apelarea subprogramului `getAdj` care are un singur argument: valoarea *C*. Primul apel al acestui subprogram returnează prima intersecție vecină intersecției *C*, al doilea apel returnează a doua intersecție vecină intersecției *C* și așa mai departe până la al M_C-lea apel care returnează cea de-a M_C-a intersecție vecină. Nu este permisă apelarea acestui subprogram decât de cel mult M_C ori, pentru o anumită intersecție *C*.

După ce programul vostru a găsit unul dintre cele mai scurte drumuri de la intersecția *S* până la intersecția *F*, el trebuie să apeleze o singură dată subprogramul `done`, cu un singur argument: numărul *R* al intersecțiilor care se află de-a lungul drumului dintre cele două intersecții (incluzând intersecțiile *S* și *F*).

În continuare, programul trebuie să apeleze subprogramul `report` exact de *R* ori, transmițând de fiecare dată câte un argument: numărul de ordine al unei intersecții (apelurile trebuie efectuate în ordinea în care apar intersecțiile de-a lungul drumului). Ulterior, programul trebuie să se oprească.

După apelarea subprogramului `done`, nu se poate apela nici un alt subprogram cu excepția subprogramului `report`.

Pentru fiecare test programul va fi punctat în funcție de numărul total *K* al apelurilor subprogramelor `getXYM` și `getAdj`.



Acest număr va fi comparat cu numărul *J* de apeluri ale acelorași subprograme de către programul juriului care rezolvă aceeași problemă.

Dacă valoarea *K* este cel mult egală cu valoarea *J*, atunci vor fi acordate 10 puncte pentru testul respectiv.

În caz contrar, vor fi acordate $2 + 4 \cdot (T - K) / (T - J)$ puncte, unde *T* este numărul total de intersecții și străzi.

Dacă programul nu găsește unul dintre cele mai scurte drumuri, dacă încalcă regulile de folosire a bibliotecii, sau dacă răspunde corect accidental fără să asigure corectitudinea (de exemplu, dacă programul definește una din cele mai scurte căi fără nici un apel al subprogramelor `getXYM` și `getAdj`), va fi punctat cu 0 puncte pentru testul respectiv.

Testele sunt formate din date reale, oferite cu bunăvoință de *DATECS GIS Center*.

Biblioteca Pascal

```
procedure start(var N,S,F:
                        LongInt);
procedure getXYM(I: LongInt;
      var XI,YI,MI: LongInt);
function getAdj(I: LongInt):
                        LongInt;
procedure done(R: LongInt);
procedure report(V: LongInt);
```

Biblioteca C/C++

```
void start(long* N, long* S,
                    long* F);
void getXYM(long I, long* XI,
        long* YI, long* MI);
long getAdj(long I);
void done(long R);
void report(long V);
```

Restricții

Numărul intersecțiilor din oraș este cuprins între 10 și 7000.

Coordonatele intersecțiilor sunt numere întregi cuprinse între 0 și 10.000.

Fiecare intersecție are cel mult cinci vecini, iar numărul total al străzilor este de cel mult 16.000.

Experimente

Biblioteca citește harta orașului din fișierul **`path.in`**. Primul rând al acestui fișier va conține numerele întregi *N*, *S* și *F*, separate prin câte un spațiu.

Fiecare dintre următoarele *N* linii conține descrierea unei intersecții, în ordinea numerelor de ordine ale intersecțiilor. Primele trei numere ale liniei vor fi X_C, Y_C și M_C. Ele vor fi urmate de M_C numere care reprezintă numerele de ordine ale intersecțiilor vecine.

În cazul în care programul folosește biblioteca corect și dacă programul returnează un drum valid, atunci drumul va fi descris în fișierul **`path.out`**.

Exemple

`path.in`
```
3 1 3
0 0 1 2
0 1 1 3
1 1 0
```

`path.out`
```
3
1 2 3
```

Apel	*Rezultat*
`start(N,S,F)`	`N=3 S=1 F=3`
`getXYM(1,...)`	$X_1 = 0$ $Y_1 = 0$ $M_1 = 1$
`getXYM(3,...)`	$X_3 = 1$ $Y_3 = 1$ $M_3 = 0$
`getAdj(1)`	`2`
`getXYM(2,...)`	$X_2 = 0$ $Y_2 = 1$ $M_2 = 1$
`getAdj(2)`	`3`
`done(3)`	
`report(1)`	
`report(2)`	
`report(3)`	

`path.in`
```
4 1 4
0 0 2 2 3
0 1 1 4
9 0 2 1 4
1 1 1 3
```

`path.out`
```
3
1 2 4
```

Apel	*Rezultat*
`start(N,S,F)`	`N=4 S=1 F=4`
`getXYM(1,...)`	$X_1 = 0$ $Y_1 = 0$ $M_1 = 2$
`getXYM(4,...)`	$X_4 = 1$ $Y_4 = 1$ $M_4 = 1$
`getAdj(1)`	`2`
`getAdj(1)`	`3`
`getXYM(2,...)`	$X_2 = 0$ $Y_2 = 1$ $M_2 = 1$
`getXYM(3,...)`	$X_3 = 9$ $Y_3 = 0$ $M_3 = 2$
`getAdj(2)`	`4`
`done(3)`	
`report(1)`	
`report(2)`	
`report(4)`	

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

DATECS GIS CENTER

P040616: Cursă

Primarul orașului *Plovdiv* a decis să organizeze o cursă pe străzile orașului pentru a demonstra că acestea sunt potrivite pentru viteze mari. El trebuie să aleagă o rută pentru concurs care să fie cât mai rapidă cu putință.

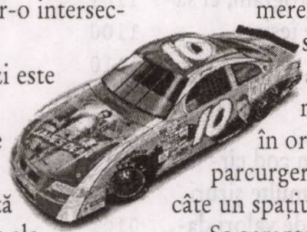
După ce a vorbit cu consilierii săi el a stabilit următoarele restricții. Cursa trebuie să înceapă și să se termine în intersecția unde se află primăria. Singurele schimbări de direcție acceptate de-a lungul traseului sunt cele spre stânga (a face o schimbare de direcție spre stânga față de direcția curentă înseamnă să schimbăm direcția cu un unghi mai mare sau egal cu 0, dar strict mai mic ca 180 de grade).

Dacă există mai multe rute ce îndeplinesc condițiile de mai sus, ruta aleasă va fi aceea pentru care cea mai scurtă stradă a rutei este cât mai lungă posibil.

Orașul *Plovdiv* are N intersecții între care se află un număr total de M străzi pe care se poate circula în ambele sensuri.

Intersecțiile sunt descrise prin două coordonate în plan și sunt numerotate de la 1 la N în ordinea în care sunt date la intrare.

Primăria se află situată în intersecția cu numărul 1. Străzile sunt linii drepte care încep într-o intersecție și se termină în alta.



Lungimea unei străzi este egală cu distanța euclidiană dintre intersecțiile pe care le unește.

O stradă este definită prin numerele de ordine ale intersecțiilor pe care le unește. Nu există mai mult de o stradă între oricare două intersecții ale orașului. Străzile nu se pot intersecta decât la capete.

Scrieți un program care determină o rută, dintre cele posibile în oraș, care începe și se termină cu intersecția 1, iar din fiecare intersecție aflată pe această rută se merge fie înainte, fie se schimbă direcția spre stânga.

Dacă există mai multe astfel de rute, va trebui aleasă cea a cărei cea mai scurtă stradă are lungimea cea mai mare. Ruta aleasă nu poate trece de două ori în același sens pe aceeași stradă.

Date de intrare

Datele se citesc de la intrarea standard.

Prima linie conține două numere întregi N și M, separate printr-un spațiu. Acestea reprezintă numărul intersecțiilor din oraș, respectiv numărul total al străzilor din *Plovdiv*.

Fiecare din următoarele N linii conține coordonatele X și Y ale unei intersecții, separate printr-un spațiu.

Pe ultimele M linii sunt descrise străzile. Fiecare dintre aceste linii conține două numere separate printr-un spațiu, acestea reprezentând numerele de ordine ale intersecțiilor unite de o stradă.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Prima linie va conține numărul de intersecții din rută (intersecția în care se află primăria fiind numărată de două ori).

Următoarea linie va conține numerele de ordine ale intersecțiilor aflate pe ruta aleasă (începând și terminând cu intersecția 1), în ordinea corespunzătoare parcurgerii rutei, separate prin câte un spațiu.

Se garantează faptul că există întotdeauna cel puțin o soluție a problemei.

Dacă există mai multe soluții, poate fi aleasă oricare dintre ele.

Restricții

Numărul intersecțiilor din oraș este cuprins între 3 și 2000.

Numărul străzilor din oraș este cuprins între 5 și 25.000.

Coordonatele intersecțiilor sunt numere întregi cuprinse între -10.000 și 10.000.

Exemplu

Intrarea standard

```
5 6
1 0
2 1
1 1
0 1
1 2
1 2
2 5
1 4
5 4
2 3
4 3
```

Ieșirea standard

```
5
1 2 5 4 1
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

P040617: Joc

Lui *Ivan* îi place foarte mult să se joace în timpul liber. Din nefericire, el nu se poate bucura întotdeauna de compania prietenilor și câteodată se plictisește fiind singur.

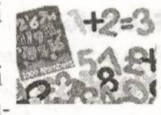
Astfel, *Ivan* inventează jocuri, în care este prezent un singur jucător. El este mândru de ultimul său joc și ar vrea să vă povestească și vouă despre el.

Se dau două șiruri finite de numere întregi pozitive. Jocul constă din mișcări consecutive. Este permisă următoarea mișcare: se șterg ultimele K_1 numere din primul șir (posibil tot șirul) calculându-se suma lor S_1, și ultimele K_2 numere din al doilea șir (posibil tot șirul) calculându-se suma lor S_2. Apoi se determină costul mișcării ca fiind $(S_1 - K_1) \cdot (S_2 - K_2)$.

Mișcările se repetă până la eliminarea tuturor numerelor din ambele șiruri. Costul total al jocului este suma costurilor tuturor mișcărilor. Scopul vostru este de a obține un cost total minim.

Nu este permisă eliminarea tuturor elementelor unui șir, în

timp ce celălalt șir mai conține elemente.



Va trebui să scrieți un program care calculează cel mai mic cost total al jocului prezentat.

Date de intrare

Datele se citesc de la intrarea standard.

Prima linie conține două numere întregi L_1 și L_2, separate printr-un spațiu, care reprezintă lungimile celor două șiruri.

Cea de-a doua linie conține L_1 numere întregi separate prin câte un spațiu, care reprezintă elementele primului șir.

Cea de-a treia linie conține L_2 numere întregi separate prin câte un spațiu, care reprezintă elementele celui de-al doilea șir.

Date de ieșire

Datele de ieșire vor fi scrise la ieșirea standard.

Va fi scrisă o singură linie pe care se va afla un singur număr care reprezintă costul minim care poate fi obținut pentru jocul descris.

Restricții

La fiecare pas, din fiecare șir va fi șters cel puțin un număr.

Lungimile celor două șiruri sunt cuprinse între 1 și 2000.

Valorile elementelor din cele două șiruri sunt cuprinse între 0 și 1000.

Exemplu

Intrarea standard

```
3 2
1 2 3
1 2
```

Ieșirea standard

```
2
```

Timp de execuție: 1 secundă/test
Memorie disponibilă: 64 MB

P040618: Coduri

Vom defini distanța $d_H(X, Y)$ dintre două șiruri X și Y de lungimi egale ca fiind distanța *Hamming*. Aceasta reprezintă numărul de poziții în care șirurile X și Y diferă. De exemplu, $d_H(1001, 0010) = 3$ și $d_H(1001111, 0010101) = 4$.

Considerăm un număr întreg strict pozitiv n. Presupunem că $C = W_1, W_2, \ldots, W_M$ este o listă de M șiruri binare de lungime n.

Considerăm că C este o listă circulară și vom defini distanța $d_C(W_i, W_j)$ dintre două șiruri W_i și W_j din listă ca fiind $d_C(W_i, W_j) = \min\{abs(i - j), M - abs(i - j)\}$.

Fie k un număr strict pozitiv mai mic decât n. Vom spune că C este un cod circular de lungime n și anvergură k dacă pentru orice pereche de numere i și j cuprinse între 1 și M sunt respectate fiecare dintre următoarele două condiții:

- dacă $d_C(W_i, W_j) \le k$, atunci $d_H(W_i, W_j) = d_C(W_i, W_j)$;
- dacă $d_C(W_i, W_j) > k$, atunci $d_H(W_i, W_j) > k$.

Problema pricipală în studiul codurilor circulare este determinarea numărului maxim de șiruri dintr-un cod circular de lungime n și anvergură k.

Valoarea exactă a acestui număr este cunoscută doar pentru câteva valori mici ale parameterilor n și k.

Pentru această problemă nu va trebui să creați nici un program, ci să generați doar fișierele de ieșire corecte.

Date de intrare

Urmează să construiți un cod circular conținând cât mai multe șiruri posibile, pentru perechi de valori date ale parametrilor n, k.

Perechile de valori pentru care trebuie determinat codul circular sunt prezentate în figura 1.

Date de ieșire

Trebuie create zece fișiere conținând codurile construite utilizând perechile de parametri din figura 1.

Prima linie a unui fișier trimis trebuie să conțină textul `#FILE code t`, unde t este numărul testului.

Următoarele M linii vor conține șiruri consecutive ale codului de lungime n și anvergură k construit.

Modalitatea de punctare

Pentru fiecare test, cea mai bună dintre toate soluțiile concurenților va primi 10 puncte.

Dacă cea mai bună soluție este un cod format din B șiruri și un concurent trimite o soluție corectă cu M șiruri, atunci punctajul primit va fi de $10 \cdot M / B$ puncte.

Punctajul pentru fiecare test va fi rotunjit astfel încât să conțină cel mult o cifră după virgulă.

Evident, dacă fișierul nu este corect, nu se va acorda nici un punct pentru testul respectiv.

Punctajul total va fi rotunjit la cel mai apropiat întreg.

Exemplu

În cele ce urmează vom considera că pentru testul 0 al problemei avem $n = 4$ și $k = 1$.

O posibilă soluție, pentru care avem $M = 8$, este:

```
#FILE code 0
1101
1100
1110
1010
1011
0011
0001
0101
```

Test #	1	2	3	4	5	6	7	8	9	10
n	5	6	6	7	7	8	8	9	10	10
k	1	1	2	1	2	1	2	2	2	3

Figura 1

Notă

Enunțurile tuturor problemelor prezentate în cadrul acestui număr al revistei au fost traduse de către membrii redacției *GInfo* și au fost modificate pentru a le mări claritatea.

PLEDOARIE pentru instruirea ASISTATĂ de CALCULATOR

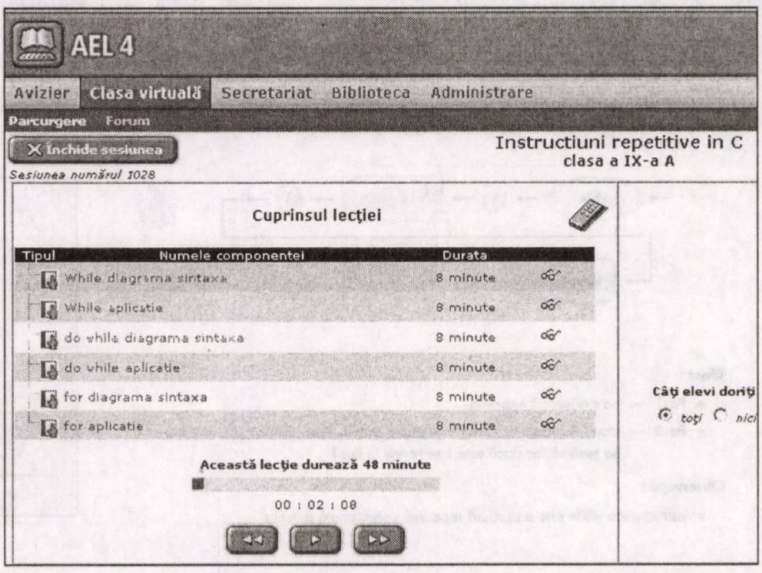
Pe parcursul unei serii de articole vă vom prezenta modul în care poate fi realizată proiectarea didactică a lecției "Instrucțiuni repetitive C++" folosind sistemul AEL pentru predare.

Situându-l pe cel care învață în centrul procesului didactic, pedagogia modernă, bazată pe interactivitate, se află într-o continuă căutare de soluții de proiectare și de aplicare de sisteme de formare care să autorizeze mai multă acțiune și interacțiune în învățare.

Instruirea asistată de calculator este o formă interactivă și cvasiautonomă de acces la cunoaștere prin medierea profesorului (doar cu îndrumarea lui), bazată pe demersurile de informare și prelucrare a informațiilor, pe o învățare în ritm propriu, realizată practic, prin observare, experimentare și descoperire, grație utilizării calculatorului și tehnologiei *multimedia*.

Platforma *e-learning AEL*, instalată pe calculatoarele laboratorului informatizat din licee, prin *soft*-urile educaționale incluse, permite realizarea unei educații multimodale, adaptate diferitelor profiluri intelectuale ale elevilor, pune elevul în situații de interacțiune mult intensificate.

Proiectarea didactică a unei lecții asistate de calculator are la bază tehnologii multimedia integrate în procesul pedagogic, în contextul strategiilor de instruire imaginate și urmărește facilitatea obținerii rezultatelor specifice învățării.

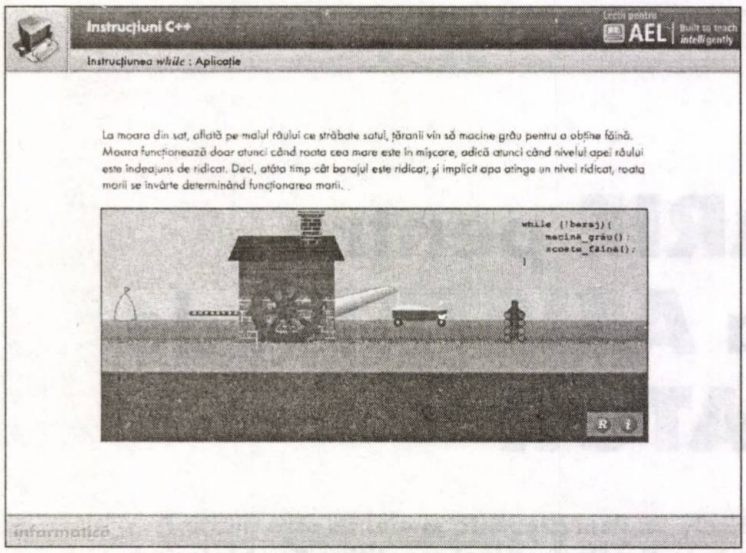


Lecția *Instrucțiuni repetitive C++* este o temă a unității de învățare *Instrucțiuni C++*, din curricula clasei a IX-a, profil real, specializarea matematică-informatică, intensiv informatică, disciplina Informatică și este structurată pe momente care definesc instructiunile *while*, *do-while* și *for*. Fiecare moment este însoțit de câte o aplicație cu ajutorul căreia elevii pot identifica situațiile de ciclare și modul în care ***instrucțiunile*** se pot folosi.

Trebuie specificat faptul că momentele - componente ale lecției - au o structură unitară pentru a crea un mediu plăcut de învățare.

Proiectarea didactică a lecției debutează prin prezentarea

- obiectivelor educaționale ale lecției: cognitive, afective si psihomotorii
- obiectivelor operaționale înscrise în manualul profesorului inclus în *soft*-ul educațional;
- strategiilor didactice:
  - ♦ principii didactice (principiul participării și învățării active, principiul conexiunii inverse);
  - ♦ metode de învățământ (metode de comunicare: expunere, conversație, problematizare; metode de acțiune: exercițiul, învățarea prin descoperire);



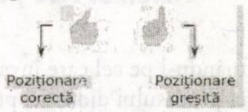
- procedee de instruire (explicația în etapa de comunicare, învățarea prin descoperire, prin rezolvarea de sarcini de lucru, problematizarea prin crearea situațiilor problemă, conversația în etapa de fixare a cunoștințelor);
- forme de organizare (frontală, individuală);
- forme de dirijare a învățării (dirijată de profesor sau independentă);
- resurse materiale (calculatoarele din laboratorul informatizat și *soft*-ul educațional inclus).

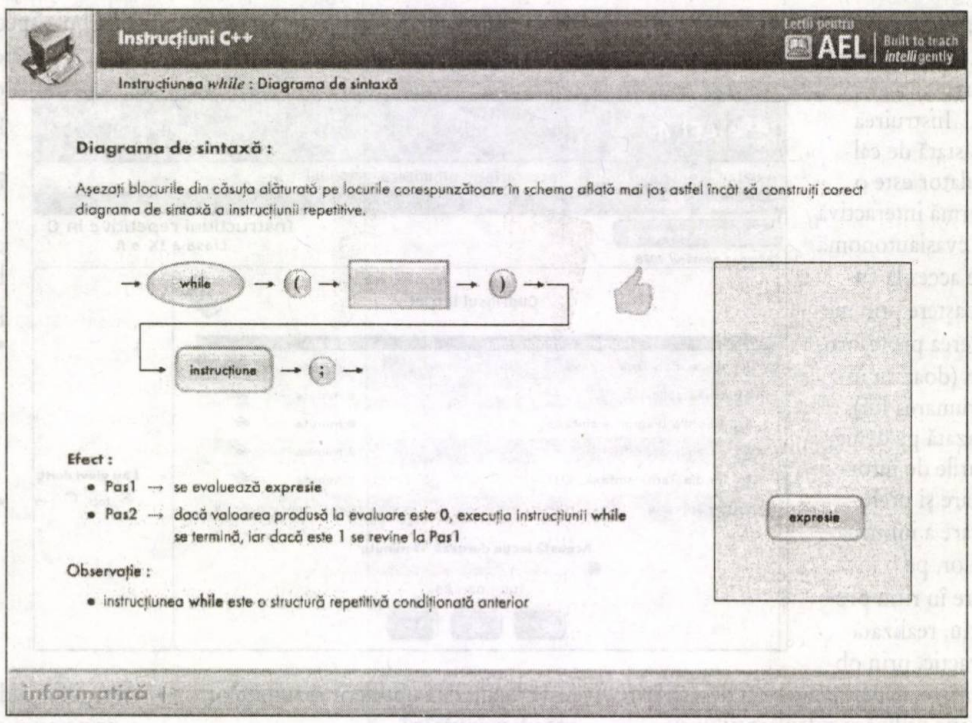
În etapa de reactualizare a cunoștințelor teoretice se vor formula întrebări referitoare la structurile repetitive studiate la elaborarea algoritmilor în pseudocod.

În primul moment al lecției intitulat *Instrucțiunea while* este dat ca exemplu modul de funcționare al unei mori. Pentru înțelegerea aplicației elevii instrumentează situații și contexte favorabile de achiziții: vor conștientiza faptul ca odată început un ciclu se poate repeta la infinit dacă nu se intervine printr-o condiție de oprire. Pentru a scădea sau ridica nivelul apei râului, elevii vor coborî sau ridica barajul, prin apăsare cu mouse-ul pe acesta. Dacă nivelul apei este ridicat, atunci moara funcționează și deci se poate măcina grâu: trăgând cu *mouse*-ul sacii cu grâu pe banda rulantă a morii se observă obținerea făinii, care va curge într-un cărucior situat pe partea cealaltă a morii. Dacă nivelul scăzut al apei nu permite acționarea roții, moara nu va funcționa.

Profesorul introduce noțiunea de diagramă de sintaxă a instrucțiunii *while*, o corelează cu teoria, iar elevul poate cerceta, descoperi, sub îndrumarea profesorului, care este forma corectă a diagramei, prin completarea schemei cu elementele aflate în partea dreaptă a ferestrei de lucru. Procedeul de descriere al diagramei este finalizat în momentul în care ultimul element al schemei a fost fixat.

Este un punct important, deoarece elevul nu poate finaliza schema diagramei de sintaxă decât dacă parcurge corect pașii ceruți, la fiecare acțiune incorectă fiind atenționat.





Acestă atenționare, repetată de altfel la construirea diagramelor altor instrucțiuni C++, are la bază o strategie elementară folosită pentru acei elevi care au nevoie de asistența profesorului.

(va urma)

Cluj-Napoca, septembrie 2004

Etapa FINALĂ

Cea de-a V-a ediție a concursului de programare organizat de GInfo s-a încheiat cu runda finală care s-a desfășurat pe data de 25 septembrie la Liceul Teoretic "Emil Racoviță" din Cluj-Napoca. În cadrul acestui articol vom prezenta câteva detalii referitoare la desfășurarea acestui concurs.

Participanți

Primii clasați în urma desfășurării celor 50 de runde ale primei faze a concursului *Bursele Agora* au fost invitați la etapa finală a acestuia.

Concurenții care s-au prezentat la faza finală a competiției au fost:

- *Mugurel-Ionuț Andreica*
- *Adrian Cârcu*
- *Mircea Pașoi*
- *Silviu-Ionuț Gănceanu*
- *Marius Dumitran*
- *Liviu Ciortea*
- *Mihail-Cosmin Pițigoi-Rada*

Competiția a fost foarte puternică, deoarece printre concurenți s-au aflat trei dintre reprezentanții *României* la concursurile internaționale din acest an.

Silviu-Ionuț Gănceanu a obținut singura medalie de aur a *României* din acest an. Mai mult, aceasta a fost obținută la *IOI* 2004, cel mai important dintre aceste concursuri.

Alături de *Mircea Pașoi* și *Marius Dumitran*, el a reprezentat *România* și la *BOI* 2004, unde a obținut o medalie de bronz. La acest concurs *Mircea Pașoi* a obținut medalia de argint.

Dintre ceilalți concurenți, *Adrian Cârcu* și *Mugurel-Ionuț Andreica* au participat la numeroase concursuri internaționale în anii anteriori.

Primul dintre ei a obținut o medalie de aur la *BOI* '99 și una de bronz la *IOI* '99, în timp ce al doilea a obținut o medalie de argint la *IOI* 2001.

Organizare

Pentru etapa finală a acestei ediții, comisia a fost formată din directorul revistei *GInfo*, *d-na lect. univ. dr. Ctara Ionescu*, directorul concursului *Bursele Agora*, *dl. Mihai Scorțaru* și reprezentantul editurii *Agora Media*, *dl. Mircea Sabău*.

Cele patru probleme propuse spre rezolvare la acest concurs au fost formulate de *Mihai Scorțaru*, *Claudiu Soroiu* și *Cosmin-Silvestru Negrușeri*.

2 1 3

Desfășurarea concursului

Concursul propriu-zis a avut loc la *Liceul Teoretic "Emil Racoviță"* din *Cluj-Napoca*, care a pus la dispoziția concurenților calculatoare dotate cu procesoare *Intel Pentium* 4 cu frecvența de 1,5 GHz. Mulțumim pe această cale *d-lui prof. Lucian Ilea* pentru sprijinul acordat în organizare.

Pentru rezolvarea celor patru probleme, concurenții au avut la dispoziție trei ore.

Imediat după terminarea concursului a început evaluarea soluțiilor concurenților. A fost folosit programul de evaluare *GInfo Evaluator*.

Câștigătorii

Concursul a fost câștigat de către *Silviu-Ionuț Gănceanu*, favoritul concursului pe baza rezultatelor din acest an. El a obținut bursa în valoare de 50$, oferită de editura *Agora Media*.

Pe poziția a doua s-a clasat *Mircea Pașoi*, care a obținut o bursă în valoare de 30$ pe lună.

Cei doi au fost urmați de către *Mugurel-Ionuț Andreica*, câștigătorul edițiilor 2000/2001 și 2001/2002 ale concursului nostru.

Fiecare contează!

După concurs, au avut loc discuții între organizatori și concurenți cu privire la posibilitățile de a îmbunătăți desfășurarea concursului.

Unele dintre propunerile finaliștilor se regăsesc în regulamentul de desfășurare al ediției 2004/2005 a competiției.

La propunerea concurenților s-a stabilit ca durata unei runde să fie fixă (24 de ore) pentru a acorda șanse egale tuturor participanților. Din același motiv s-a stabilit ca ora de începere a rundelor să fie în timpul serii.

Un nou început

Așadar, o nouă ediție a concursului nostru de programare s-a încheiat. Astfel, *Bursele Agora* a devenit cel mai longeviv concurs *online* de programare organizat în *România*.

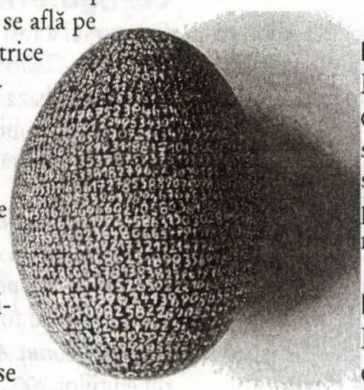
Vă așteptăm la noua ediție care începe spre sfârșitul acestui an și vă urăm mult succes!

Bursele AGORA

Vă prezentăm în continuare enunțurile celor patru probleme propuse spre rezolvare la etapa finală a ediției 2003/2004 a concursului de programare organizat de revista noastră.

P040619: Cifre

Se consideră un șir de cifre cuprinse între 1 și 9. Acest șir se află pe prima linie a unei matrice pătratice A, având dimensiunea egală cu lungimea șirului.



Următoarea linie a matricei A se obține prin permutarea la stânga cu o poziție a elementelor de pe prima linie.

Aceeași operație se aplică și pentru următoarele linii: linia i se obține prin permutarea la stânga cu o poziție a elementelor de pe linia i - 1.

În continuare se generează o matrice B care conține liniile matricei A ordonate descrescător în funcție de numărul obținut prin citirea cifrelor de la stânga la dreapta.

De exemplu, dacă șirul de caractere considerat este 2457, atunci matricea A va avea următoarea structură:

```
2457
4572
5724
7245
```

Ordonând descrescător liniile matricei A, se obține matricea B:

```
7245
5724
4572
2457
```

Se cunosc cifrele de pe ultima coloană a matricei B și se cere determinarea cifrelor de pe prima linie a acesteia.

Date de intrare

Fișierul de intrare **CIFRE.IN** conține o singură linie pe care se află cifrele aflate pe ultima coloană a matricei B.

Date de ieșire

Fișierul de ieșire **CIFRE.OUT** va conține o singură linie pe care se vor afla cifrele de pe prima linie a matricei B.

Restricții și precizări

- șirul inițial conține cel mult 10000 de cifre;
- pe baza ultimei coloane se va putea determina întotdeauna prima linie.

Exemplu

CIFRE.IN

```
5427
```

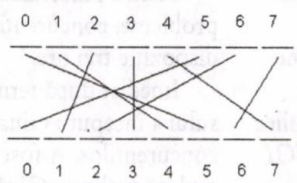
CIFRE.OUT

```
7245
```

Timp de execuție: 1 secundă / test

P040620: Produse

Se consideră o matrice pătratică A de dimensiune N ale cărei elemente sunt date de formula $a_{ij} = i^u \cdot j^v$.



Un produs va fi obținut prin alegerea unei permutări p a mulțimii $\{1, 2, \ldots, N\}$ și înmulțirea tuturor elementelor de forma a_{ix}, unde $x = p_i$.

Va trebui să determinați numărul de cifre, scris în baza 10, al rezultatului însumării tuturor produselor distincte posibile.

Două produse se consideră a fi distincte dacă permutările corespunzătoare sunt diferite.

Date de intrare

Fișierele de intrare sunt deschise, iar denumirile acestora sunt de forma **PRODUSEXX.IN**, unde XX ia valori între 01 și 20 și reprezintă numărul testului.

Fiecare dintre cele 20 de fișiere de intrare conține o singură linie pe care se află valorile N, u și v.

Date de ieșire

Pentru această problemă nu va trebui să trimiteți un program care să o rezolve, ci doar cele 20 de fișiere de ieșire corecte.

Acestea vor fi denumite **PRODUSEXX.OUT**, unde XX ia valori între 01 și 20 și reprezintă numărul testului.

Prima linie a unui fișier de ieșire trebuie să conțină textul PRODUSE TEST #XX, unde XX reprezintă numărul testului.

Cea de-a doua linie va conține numărul cifrelor sumei tuturor produselor posibile.

Exemplu

Vom considera că acest exem-

plu reprezintă testul 00.

PRODUSE00.IN

```
2 2 1
```

PRODUSE00.OUT

```
PRODUSE TEST #00
2
```

Modalitatea de acordare a punctajului

Pentru fiecare fișier de ieșire corect veți obține un anumit număr de puncte.

Punctajul maxim care poate fi obținut (dacă toate cele 20 de fișiere de ieșire sunt corecte) este de 100 de puncte.

Pentru fiecare dintre primele patru teste (**01 - 04**) se vor acorda câte trei puncte, pentru următoarele patru (**05 - 08**) câte patru puncte, pentru următoarele patru (**09 - 12**) câte cinci puncte, pentru următoarele patru (**13 - 16**) câte șase puncte, iar pentru ultimele patru (**17 - 20**) câte șapte puncte.

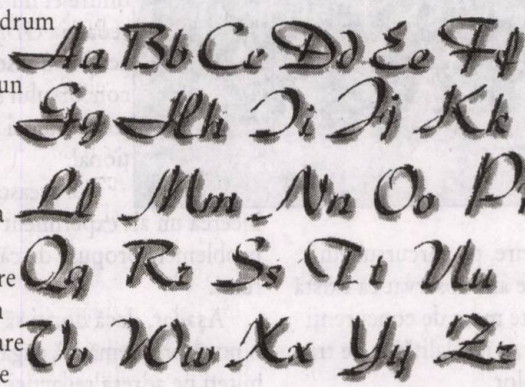
P040621: Drum

Se consideră un graf orientat aciclic care conține N noduri și M arce.

Nodurile sunt identificate prin litere distincte ale alfabetului englezesc (sunt permise atât litere mari, cât și litere mici).

Un drum este format dintr-o succesiune de arce, iar aceste drumuri sunt ordonate în ordine lexicografică.

Fiecărui drum al grafului îi corespunde un șir de caractere obținut prin concatenarea literelor corespunzătoare nodurilor, în ordinea în care sunt parcurse nodurile pentru obținerea drumului.



Se consideră că, din punct de vedere lexicografic, toate literele mici sunt înaintea tuturor literelor mari.

Dintre toate drumurile din graf, sunt luate în considerare doar cele mai lungi (lungimea unui drum este dată de numărul de arce care îl formează).

Va trebui să scrieți un program care, dintre cele mai lungi drumuri, îl determină pe cel de-al k-lea în ordine lexicografică.

Date de intrare

Prima linie a fișierului de intrare **DRUM.IN** conține trei numere întregi N, M și k separate între ele printr-un singur spațiu.

Fiecare dintre următoarele M linii conține două caractere ale alfabetului englezesc c_1 și c_2, separate între ele printr-un singur spațiu, cu semnificația că există un arc care pleacă din nodul identificat prin litera c_1 și ajunge în nodul identificat prin litera c_2.

Date de ieșire

Fișierul de ieșire **DRUM.OUT** va conține pe o singură linie un șir de caractere care reprezintă drumul determinat. Caracterele acestui șir nu vor fi separate între ele prin spații.

Restricții și precizări

- numărul de drumuri de lungime maximă este mai mare sau egal decât k;
- $2 \le N \le 52$;
- $1 \le M \le N \cdot (N - 1) / 2$;
- între oricare două noduri poate exista cel mult o muchie.

Exemplu

DRUM.IN

```
3 3 1
a b
b c
a c
```

DRUM.OUT

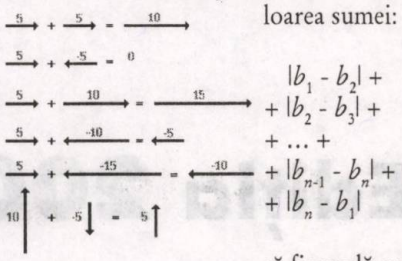
```
abc
```

Timp de execuție: 0.5 secunde/test

P040622: Numere

Se consideră un șir a format din n numere întregi, precum și un număr întreg S.

Va trebui să determinați o permutare b a șirului a astfel încât valoarea sumei:



$$|b_1 - b_2| + |b_2 - b_3| + \ldots + |b_{n-1} - b_n| + |b_n - b_1|$$

să fie egală cu S, unde b_i reprezintă cel de-al i-lea element al permutării care a fost determinată.

Cu alte cuvinte, va trebui să determinați o permutare pentru care suma valorilor absolute ale diferențelor elementelor aflate pe poziții consecutive este dată.

Date de intrare

Prima linie a fișierului de intrare **NUMERE.IN** va conține două numere întregi n și S (separate printr-un spațiu).

Acestea reprezintă lungimea șirului a, respectiv suma precizată în enunț.

Cea de-a doua linie conține cele n elemente ale șirului a separate între ele prin câte un spațiu.

Date de ieșire

Prima linie a fișierului de ieșire **NUMERE.OUT** va conține cele n elemente ale șirului b separate între ele prin câte un spațiu.

Restricții și precizări

- $1 \le N < 20$;
- $-1.000.000 \le a_i < 1.000.000$;
- va exista întotdeauna cel puțin o soluție;
- dacă există mai multe soluții va trebui să furnizați doar una dintre acestea.

Exemplu

NUMERE.IN

```
4 12
5 2 4 7
```

NUMERE.OUT

```
2 5 4 7
```

Timp de execuție: 2 secunde/test

Ediția 2004/2005

O nouă ediție a concursului nostru va începe în curând. Ca de obicei, scopul concursului este pregătirea elevilor și studenților în vederea participării la concursurile naționale și internaționale de informatică. Așadar, îi invităm pe toți cei interesați să participe la această a șasea ediție a concursului organizat de GInfo.

În primul rând trebuie să vă atenționăm că au fost efectuate unele modificări importante ale regulamentului de participare, motiv pentru care vă recomandăm să îl recitiți.

Probabil cea mai importantă modificare a regulamentului constă în noua modalitate de determinare a clasamentului final.

Sistemul de credite introdus (pentru detalii consultați regulamentul) are ca scop departajarea mai clară a concurenților. Exista întotdeauna problema cauzată de gradele diferite de dificultate ale unor probleme.

Sistemul în care pentru fiecare problemă se acordau câte 100 de puncte nu era echitabil, deoarece un concurent care rezolva o problemă foarte dificilă, dar nu putea participa la o rundă la care problema era foarte ușoară, avea același punctaj total ca și numeroșii concurenți care rezolvau problema ușoară.

O altă noutate o constituie departajarea candidaților care obțin același număr de puncte pe baza timpului de care au avut nevoie pentru a rezolva problema. Această prevedere a fost introdusă deoarece o mulțime dintre concurenți sunt studenți care sa pregătesc pentru concursul *ACM*.

Nu dorim să organizăm un concurs care să respecte în totalitate regulile *ACM*, dar considerăm că această nouă prevedere va face ca *Bursele Agora* să devină o competiție mai interesantă.

Așa cum am precizat și în articolul dedicat prezentării desfășurării etapei finale a ediției trecute, unele propuneri ale concurenților se regăsesc în noul regulament.

Așteptăm în continuare propunerile voastre pentru modificarea prevederilor regulamentului.

Din nefericire, pe parcursul edițiilor anterioare am observat că există un număr foarte mare de concurenți care nu respectau modalitățile de trimitere a soluțiilor.

Deși în cadrul ediției 2003/2004 am încercat să determinăm concurenții să respecte structura arhivelor trimise, problemele nu s-au rezolvat în totalitate. Ca urmare, regulamentul acestei ediții prevede o penalizare de 10% a punctajului în cazul în care se efectuează de către redacție orice modificare asupra unei arhive primite de la concurenți.

În sfârșit, la cererea unor concurenți, vom încerca să înlocuim vechile compilatoare *Borland C++* și *Borland Pascal* cu mult mai performantele *FreePascal* și *GNU C/C++*.

Totuși, pentru a nu descuraja concurenții neexperimentați, noile compilatoare vor fi utilizate numai pentru a studia oportunitatea înlocuirii totale a acestora începând cu ediția a șaptea a concursului.

Datorită faptului că la ediția anterioară trei concurenți din afara *României* s-au calificat la faza finală, dar nici unul dintre ei nu a participat, redacția *GInfo* a decis ca această a șasea ediție a concursului să se desfășoare numai la nivel național.

La această ediție vom încerca un alt experiment și anume problemele propuse de către concurenți.

Așadar, dacă doriți să propuneți o nouă problemă, vă rugăm să o trimiteți pe adresa redacției.

Un concurent care va trimite o poblemă, nu va putea trimite o soluție la problema respectivă, dar va primi un număr de credite care va fi stabilit de către redacție, în funcție de "calitatea" problemei respective.

REGULAMENT de participare

Ediția a VI-a a concursului de programare organizat de Gazeta de Informatică începe pe data de 15 decembrie 2004. Pentru detalii referitoare la regulamentul de desfășurare al acestui concurs, vă invităm să citiți paginile următoare.

Informații generale

Concursul de programare *Bursele Agora* este o competiție destinată elevilor și studenților. Acest concurs este organizat de redacția revistei *GInfo* și editura *Agora Media*.

Prima fază a concursului constă în 50 de runde; la fiecare dintre ele participanții trebuie să rezolve câte o problemă și să trimită soluțiile prin *e-mail*; adresa de *e-mail* a concursului nostru este **concurs@ginfo.ro**.

După etapele care se vor desfășura prin *e-mail*, va avea loc o etapă finală la *Cluj-Napoca*; doar primii zece clasați în urma primei faze a concursului vor participa la faza finală. Cheltuielile de cazare și transport pe teritoriul *României* vor fi suportate de organizatori.

Problemele care trebuie rezolvate sunt propuse de membrii redacției *GInfo* în colaborare cu studenți și profesori care se ocupă de pregătirea elevilor români care participă la diferite concursuri naționale și internaționale.

Premii

Concurentul clasat pe prima poziție în urma desfășurării fazei finale va fi premiat cu o bursă în valoare de 50$ pe lună, pe durata anului școlar sau a anului universitar următor.

Concurentul clasat pe a doua poziție va fi premiat cu o bursă în valoare de 30$ pe lună, pe durata anului școlar/universitar următor.

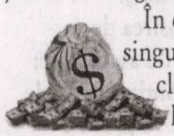
Așadar, cuantumul total al premiilor este de 80$ pe lună. La aceste premii se vor adăuga altele oferite de diferiți sponsori.

Concursul de programare Bursele Agora este o competiție destinată elevilor și studenților. Acest concurs este organizat de redacția revistei GInfo și editura Agora Media.

Prima fază a concursului constă în 50 de runde; la fiecare dintre ele participanții trebuie să rezolve câte o problemă și să trimită soluțiile prin e-mail; adresa de e-mail a concursului nostru este concurs@ginfo.ro.

După etapele care se vor desfășura prin e-mail, va avea loc o etapă finală la Cluj-Napoca; doar primii zece clasați în urma primei faze a concursului vor participa la faza finală.

În cazul în care doi sau mai mulți concurenți se vor afla la egalitate pe primul loc, suma de 80$ va fi împărțită în mod egal între aceștia.



În cazul în care un singur concurent se clasează pe primul loc, dar doi sau mai mulți concurenți se află la egalitate pe locul al doilea, atunci suma de 30$ va fi împărțită în mod egal între aceștia din urmă, primul clasat fiind premiat cu 50$.

În cazul în care un concurent se clasează pe o poziție care îi dă dreptul la o bursă dar, potrivit secțiunii *Dreptul de participare* a prezentului regulament, el nu are dreptul să o câștige, atunci va fi eliminat din clasament.

Organizatorii concursului își rezervă dreptul de a modifica premiile în orice moment.

Înscrierea

Pentru a participa la această ediție a concursului de programare *Bursele Agora* va trebui, mai întâi, să vă înscrieți. Cea mai simplă modalitate de înscriere este completarea *online* a formularului de înscriere (**www.ginfo.ro/concurs/inscriere.shtml**) care va fi trimis automat, prin intermediul clientului dumneavoastră de *e-mail*, la adresa **concurs@ginfo.ro**.

O altă posibilitate o constituie trimiterea unui *e-mail* la adresa **concurs@ginfo.ro**.

Acesta va trebui să conțină informațiile cerute în formularul disponibil *online* și anume: numele și prenumele dumneavoastră, data nașterii, adresa *e-mail*, adresa poștală și numărul de telefon.

Cea de-a treia posibilitate constă în trimiterea unei scrisori care va conține aceleași informații. Aceasta trebuie trimisă pe adresa:

Redacția GInfo
str. Universității nr. 8/8,
400091 Cluj-Napoca

În termen de cinci zile lucrătoare de la primirea de către redacția *GInfo* a *e-mail*-ului sau a scrisorii dumneavoastră, vă vom trimite un *e-mail* care va conține codul dumneavoastră de identificare pentru ediția 2004/2005 a concursului.

Dreptul de participare

La prima fază a concursului ***nu au drept de participare*** concurenții care au fost descalificați la una dintre precedentele patru ediții ale concursului.

De asemenea, nu au drept de participare autorii articolelor publicate în *GInfo* pe parcursul desfășurării concursului. În cazul în care este publicat un articol care are ca autor sau coautor un anumit concurent, atunci concurentul nu are dreptul să participe la concurs, începând cu ziua primirii materialului la redacție.

Fac excepție de la această regulă articolele care conțin enunțurile sau soluțiile unor probleme propuse spre rezolvare la diferite concursuri de programare.

Nu au drept de participare la concurs membrii redacției *GInfo*, angajații editurii *Agora Media*, precum și rudele de gradul I și II ale acestora.

Nu au drept de participare la *faza finală* a concursului ***Bursele Agora*** concurenții care au participat deja la cinci finale ale acestuia.

Nu au dreptul de *a câștiga* una dintre ***Bursele Agora*** concurenții care au beneficiat deja de trei burse în anii anteriori.

Desfășurarea concursului

Limbajele de programare acceptate la fiecare rundă a concursului *Bursele Agora* sunt *Pascal* și *C/C++*. Compilatoarele utilizate pentru compilarea surselor concurenților vor fi specificate în enunțul fiecărei probleme.

Aceste enunțuri vor fi publicate pe *site*-ul *Web* al revistei noastre într-o secțiune dedicată acestui scop (**www.ginfo.ro/concurs/probleme.shtml**). Rezolvările vor trebui trimise înaintea expirării termenului limită specificat pentru fiecare rundă.

Fiecare etapă va fi anunțată cu cel puțin 24 de ore înaintea momentului în care aceasta va începe. Rundele vor începe la ora 20:00 și vor dura câte 24 de ore, iar anunțul va conține data începerii rundei. Există posibilitatea ca două sau mai multe etape (cel mult zece) să se desfășoare simultan (pe parcursul aceleiași zi).

Enunțul problemei va fi disponibil începând cu data specificată în anunț, la ora 20:00. În acest moment va fi anunțat și termenul limită (ora 20:00 a zilei următoare) pentru trimiterea soluțiilor.

Termenul limită poate fi prelungit printr-o decizie a organizatorilor. În acest caz, va fi publicat un anunț, cel târziu la ora 19:00.

Nu vor fi acceptate soluțiile care ajung la redacție după expirarea termenului limită

O singură soluție va fi acceptată pentru fiecare problemă. Dacă un concurent va trimite mai multe soluții pentru o anumită problemă, numai prima va fi luată în considerare, restul fiind ignorate. Redacția *GInfo* își rezervă dreptul de a descalifica concurenții care trimit soluții identice (sau aproape identice).

Este interzisă folosirea de secvențe scrise în limbaj de asamblare; dacă un fișier sursă conține astfel de secvențe, concurentul va primi 0 puncte pentru problema de la runda respectivă.

Este interzis ca programele generate folosind sursele concurenților să modifice *timer*-ul. Concurenții care vor încerca să modifice ceasul sistem vor fi descalificați.

Dacă aveți neclarități referitoare la enunțurile problemelor, puteți adresa întrebări folosind adresa *e-mail* a concursului.

Nu puteți adresa decât întrebări ale căror răspunsuri posibile sunt DA sau NU.

Dacă răspunsul poate fi găsit în enunțul problemei sau întrebarea nu respectă condiția anterioară, atunci răspunsul la întrebare va fi FĂRĂ COMENTARII.

În cazul în care redacția *GInfo* consideră că răspunsul la o anumită întrebare trebuie primit de toți participanții, atunci cel care a adresat întrebarea va primi răspunsul ANUNȚ GENERAL.

Anunțurile generale vor fi disponibile *online* în secțiunea dedicată problemelor.

Trimiterea soluțiilor

Pentru a trimite soluțiile de la o anumită rundă va trebui să trimiteți un *e-mail* având ca subiect codul dumneavoastră de identificare urmat de textul `Runda XX`, unde `XX` reprezintă numărul rundei.

Acest *e-mail* va avea atașată o arhivă `.ZIP`, `.ACE` sau `.RAR` numită **`YYYYYRXX.EXT`**, unde `YYYYY` este codul dumneavoastră de identificare, `XX` este numărul rundei, iar `EXT` este extensia (`RAR`, `ACE` sau `ZIP`).

Arhiva va conține două fișiere. Primul dintre ele va fi un fișier `.BAT` care va conține linia de comandă pe care doriți să o folosim pentru a compila fișierul sursă (al doilea fișier al arhivei), corespunzător rundei respective. Acest fișier se va numi **`YYYYYRXX.BAT`**, unde `YYYYY` este codul dumneavoastră de identificare, iar `XX` este numărul rundei.

Cel de-al doilea fișier din arhivă va conține codul sursă corespunzător rezolvării pe care doriți să o trimiteți pentru problema de la runda respectivă. Denumirea acestuia va fi **YYYYYRXX.PAS**, **YYYYYRXX.C** sau **YYYYYRXX.CPP**. Din motive de securitate, arhivele autoexpandabile nu sunt acceptate.

Liniile de comandă din fișierul .BAT trebuie să aibă efect doar asupra fișierului sursă trimis de concurenți. Dacă fișierul conține comenzi care manipulează alte fișiere sau alte operații, atunci redacția *GInfo* își rezervă dreptul de a-l avertiza sau descalifica pe concurentul respectiv.

Arhiva nu va conține nici un alt fișier sau director, deci nu puteți trimite fișiere auxiliare împreună cu fișierele sursă.

Dacă programul dumneavoastră trebuie să creeze fișiere auxiliare, atunci acestea vor avea extensia .TMP, vor fi create în directorul curent și trebuie șterse înaintea terminării execuției programului. Dacă această regulă nu este respectată concurentul va primi 0 puncte pentru problema corespunzătoare, chiar dacă soluția este corectă.

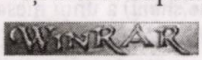
Dacă redacția *GInfo* consideră că un concurent pune în pericol buna desfășurare a competiției (prin suprascrierea unor fișiere, crearea de fișiere foarte mari etc.), atunci acel concurent poate fi descalificat.

Toate fișierele trimise devin proprietatea editurii *Agora Media* și nu pot fi folosite în scopuri comerciale fără acordul scris al editurii.

Pentru dezarhivarea fișierelor trimise de concurenți se vor folosi următoarele programe:

- *WinZip* 9.0;
- *WinRar* 3.4;
- *WinAce* 2.6.

La unele runde nu va trebui să trimiteți soluții sub forma unor coduri sursă, ci doar fișierele de ieșire corecte pentru anumite fișiere de intrare pe care le veți avea la dispoziție.



Pentru aceste runde modalitatea de trimitere a soluțiilor va fi descrisă în enunțul problemei.

În cazul în care, pentru o anumită rundă, un concurent nu respectă modalitatea de trimitere a soluțiilor, atunci redacția *GInfo* își rezervă dreptul de a modifica anumite fișiere pentru a putea testa soluția. Redacția *GInfo* nu își asumă nici o răspundere cu privire la efectele modificărilor asupra punctajului concurentului.

În cazul în care redacția *GInfo* efectuează astfel de modificări, concurentul va fi penalizat cu 10% din punctajul de la runda respectivă.

Evaluarea

Pentru fiecare problemă puteți primi maximum 100 de puncte. Pentru corectarea fiecărei probleme vom folosi între 5 și 20 de seturi de date; dacă, pentru un anumit set de date, fișierul de ieșire este corect, atunci concurentul va primi un număr de puncte care se va adăuga la numărul de puncte corespunzător problemei respective. În cazul anumitor probleme este posibilă acordarea de punctaje parțiale.

Evaluarea va fi automată; în cazul în care concurenții trebuie să trimită programe de rezolvare a problemelor, în vederea efectuării evaluării se va folosi întotdeauna ultima versiune a programului de evaluare *GInfo Evaluator*.

Calculatorul folosit pentru evaluarea soluțiilor are următoarea configurație *hardware*: procesor *Intel Pentium M* cu frecvența de 1,4 GHz și memorie *cache* de 1 MB, RAM: 256 MB, HDD: 40 GB.

Este garantat faptul că în timpul evaluării vor fi liberi cel puțin 400 KB de memorie convențională și cel puțin 100 MB de spațiu liber pe disc.

Sistemul de operare folosit în timpul evaluării va fi *Microsoft Windows XP Professional Service Pack 2*.

Dacă pentru un anumit test, un program va efectua o operație ilegală, atunci concurentul va primi 0 puncte pentru setul de date corespunzător (indiferent de tipul operației).

Pentru fiecare rundă se va stabili un clasament în funcție de numărul total de puncte obținute.

Concurenții cu același număr de puncte, vor fi departajați pe baza momentului la care redacția *GInfo* a primit soluțiile. Va fi luată în considerare ora și minutul la care a ajuns la redacție *e-mail*-ul care conține soluția.

În cazul în care punctajul, ora și minutul sunt identice, concurenții sunt considerați a fi la egalitate.

Pentru fiecare rundă vor fi acordate un număr total de 1000 de credite.

Acestea vor fi împărțite între primii clasați astfel:

- locul *I*: 100 de credite
- locul *II*: 90 de credite
- locul *III*: 80 de credite
- locul *IV*: 75 de credite
- locul *V*: 70 de credite
- locul *VI*: 65 de credite
- locul *VII*: 60 de credite
- locul *VIII*: 55 de credite
- locul *IX*: 50 de credite
- locul *X*: 45 de credite
- locul *XI*: 40 de credite
- locul *XII*: 35 de credite
- locul *XIII*: 31 de credite
- locul *XIV*: 28 de credite
- locul *XV*: 25 de credite
- locul *XVI*: 22 de credite
- locul *XVII*: 20 de credite
- locul *XVIII*: 18 credite
- locul *XIX*: 16 credite
- locul *XX*: 15 credite
- locul *XXI*: 14 credite
- locul *XXII*: 13 credite
- locul *XXIII*: 12 credite
- locul *XXIV*: 11 credite
- locul *XXV*: 10 credite.

Dacă doi concurenți se vor afla la egalitate, atunci creditele corespunzătoare pozițiilor lor vor fi împărțite în mod egal între ei. De exemplu, dacă doi concurenți se află la egalitate pe primul loc, atunci fiecare va primi câte 95 de credite, iar dacă doi concurenți se află la egalitate pe locul

XXV, atunci fiecare va primi câte 5 credite.

Clasamentul final va fi obținut prin însumarea creditelor obținute la fiecare rundă.

Sfaturi

Programatorii în *Pascal* nu ar trebui să folosească niciodată *unit*-ul `CRT`, dacă este specificat compilatorul *Borland Pascal*, deoarece programele lor se vor opri returnând eroarea de execuție `Error 200: Division by zero`. Acest fapt este cauzat de un bine cunoscut *bug* al compilatorului *Borland Pascal* 7.0; nici o problemă propusă spre rezolvare în cadrul concursului *Bursele Agora* nu necesită folosirea acestui unit.

Atunci când sunt folosite anumite tipuri de date (cum ar fi `Comp`) modul `8087` trebuie să fie activ. Opțiunile de mediu permit setarea acestui mod ca mod implicit, dar de obicei acest mod nu este activ.

Programatorii în *Pascal* sunt rugați să includă în codul sursă directiva de compilare `{$N+}`, dacă doresc să folosească tipuri de date care necesită activarea acestui mod. În caz contrar, în mometul în care vom încerca să compilăm fișierul sursă, va fi generată eroarea de compilare `Error 116: Must be in 8087 mode to compile this`.

Pentru a vă asigura că programul se va compila exact ca pe calculatorul folosit pentru a-l scrie, apăsați `CTRL+O,O`. Astfel, în sursa dumneavoastră, vor fi incluse toate directivele de compilare cu opțiunile care sunt setate în mediu.

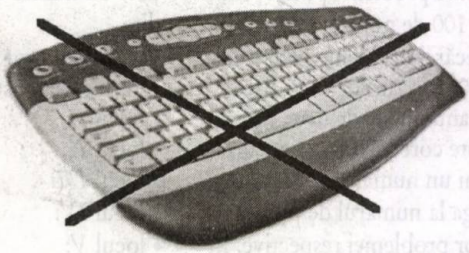
Puteți modifica aceste opțiuni astfel încât să fie cele dorite de dumneavoastră. Nu vă îngrijorați dacă nu cunoașteți semnificația anumitor directive de compilare; acestea nu vor avea efecte negative asupra execuției programului; dacă programul dumneavoastră va rula corect pe calculatorul pe care l-ați folosit pentru a-l scrie, atunci va rula corect și pe calculatorul folosit pentru evaluare.

Toate programele trebuie să citească datele de intrare din fișierul de intrare specificat. Nu ar trebui să fie așteptate date de la intrarea standard (tastatură).

Dacă programul dumneavoastră va aștepta o reacție de la tastatură, atunci execuția sa va fi oprită după expirarea timpului și nu veți obține nici un punct pentru acea soluție, chiar dacă ea este corectă.

Așadar, vă recomandăm să nu includeți în codul sursă secvențe de tipul `Readln`, `getch()`, `getche()` etc.

Toate programele trebuie să scrie datele de ieșire în fișierul de ieșire specificat.



Puteți scrie date și la ieșirea standard, dar aceasta nu va duce decât la încetinirea programului, deci nu este recomandabil să scrieți date pe ecran. Toate aceste date scrise la ieșirea standard sunt ignorate de către programul de avaluare.

Programul de evaluare va verifica numai soluția scrisă în fișierul de ieșire specificat.

Fișierul de intrare se află întotdeauna în directorul curent; nu este nevoie să încercați să citiți date din alte locații.

Fișierul de ieșire va trebui generat tot în directorul curent; nu este niciodată necesar să scrieți date de ieșire în alte locații.

Trebuie să folosiți întotdeauna denumirile specificate în enunț pentru fișierul sursă, fișierul de intrare și fișierul de ieșire.

Folosirea altor denumiri va duce la funcționarea incorectă a programului.

Exemplu

Limbajele de programare acceptate: Pascal, C și C++.

Compilatoarele utilizate: Borland Pascal 7.0 și *Borland C++* 3.1.

Descriere: Dându-se două numere, aflați suma lor.

Date de intrare: Fișierul de intrare **`SUMA.IN`** conține, pe o singură linie, cele două numere separate printr-un singur spațiu.

Date de ieșire: Fișierul de ieșire **`SUMA.OUT`** trebuie să conțină suma celor două numere.

Exemplu:

`SUMA.IN`	**`SUMA.OUT`**
2 4	6

Timp de execuție: 1 secundă/test

După citirea enunțului veți începe să rezolvați problema. Veți observa că nu este clar dacă numerele sunt întregi sau reale.

Pentru a clarifica acest lucru va trebui să adresați o întrebare. Aceasta nu poate fi: *Numerele sunt întregi sau reale?* deoarece nu se poate răspunde la ea prin `DA` sau `NU`. O soluție ar fi să întrebați: *Numerele sunt întregi?* Veți primi răspunsul `NU` și veți ști că numerele sunt reale.

Acum apare o întrebare mult mai importantă: *Ce precizie trebuie folosită pentru afișarea rezultatului?* Se pare că redacția *GInfo* a uitat să precizeze acest lucru în enunț.

Bineînțeles, va trebui să adresați o întrebare la care să se poată răspunde prin `DA` sau `NU`. Ați putea încerca: *Este suficientă o precizie de trei zecimale?*

Răspunsul la această întrebare va fi `ANUNȚ GENERAL` și va fi urmat de publicarea în secțiunea dedicată problemelor (**www.ginfo.ro/concurs/probleme.shtml**) a unui mesaj de genul:

Rezultatul va trebui scris cu o precizie de două zecimale. Ne cerem scuze pentru descrierea incompletă a

problemei.

Sperăm să nu fim nevoiți să publicăm prea des astfel de mesaje, dar...

După citirea anunțului general puteți începe scrierea codului sursă. O soluție corectă în *Pascal* ar putea fi:

```
 1: Program SUMA;
 2: var a,b:Real;
 3:     f:Text;
 4: Begin
 5:   Assign(f,'SUMA.IN');
 6:   Reset(f);
 7:   Readln(f,a,b);
 8:   Close(f);
 9:   Assign(f,'SUMA.OUT');
10:   Rewrite(f);
11:   Writeln(f,a+b:0:2);
12:   Close(f)
13: End.
```

Numerele de la începutul fiecărei linii nu fac parte din program; ele sunt folosite pentru a ne putea referi mai ușor la o anumită linie.

Dacă veți adăuga o linie `1': uses Crt;` și o linie `4': Clrscr;` nu veți reuși să obțineți decât eroarea `Error 200: Division by zero` așa cum am explicat în secțiunea ***Sfaturi***.

Dacă adăugați o linie `12': Readln;` rezultatul evaluării va fi `Time out!`, deoarece execuția programului nu se va termina, ci se va aștepta apăsarea tastei **<ENTER>**. Execuția va fi întreruptă de către programul de evaluare după o secundă (timpul maxim de execuție admis, specificat în enunțul problemei).

Dacă folosiți linia `5: Assign(f,'C:\SUMA.IN');` atunci programul va funcționa corect numai dacă directorul curent este rădăcina *drive*-ului `C:` (ceea ce este foarte improbabil). Nu veți obține decât eroarea `File not found`.

Dacă folosiți linia `5: Assign(f,'ALTFISIER');` atunci programul nu va citi datele de intrare din fișierul specificat și va rula incorect.

Dacă folosiți linia `9: Assign(f,'C:\SUMA.OUT');` atunci programul va părea că funcționează corect, dar programul de evaluare nu va găsi fișierul dumneavoastră de ieșire decât dacă directorul curent este rădăcina *drive*-ului `C:` (din nou, șansele sunt foarte mici). În acest caz, programul de evaluare va returna mesajul `No output file!`.

Dacă folosiți linia `9: Assign(f,'ALTFISIER');` atunci programul dumneavostră nu va scrie datele în fișierul de ieșire specificat, iar programul de evaluare va returna `No output file!`.

Dacă folosiți linia `7: Readln(a,b);` atunci programul va aștepta datele de intrare de la intrarea standard și va fi întrerupt după o secundă (timpul maxim de execuție specificat în enunț).

Puteți folosi această linie numai dacă aveți o secvență asemănătoare cu:

```
5: Assign(Input,'SUMA.IN');
6: Reset(Input);
7: Readln(a,b);
8: Close(Input);
```

Dacă folosiți linia `11: Writeln(a+b:0:2);` atunci datele de ieșire vor fi scrise pe ecran și nu în fișierul de ieșire specificat.

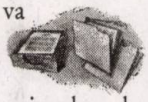
Totuși, puteți folosi această linie în cazul în care secvența este de tipul:

```
 9: Assign(Output,
                  'SUMA.OUT');
10: Rewrite(Output);
11: Writeln(a+b:0:2);
12: Close(Output)
```

Acestea sunt cele mai des întâlnite erori care au apărut la precedentele ediții ale concursului organizat de *GInfo*. Sperăm că nu veți face aceleași greșeli și la această ediție.

După scrierea codului sursă trebuie să trimiteți soluția. În cele ce urmează vom considera că *ID*-ul dumneavoastră este `02457` și participați la prima rundă a concursului.

Trebuie să creați arhiva pe care o veți trimite; aceasta se va numi **02457R01.ZIP**, **02457R01.ACE** sau **02457R01.RAR** și va conține doar două fișiere: **02457R01.BAT**, respectiv **02457R01.PAS**.



Fișierul **02457R01.PAS** va conține codul sursă al rezolvării, în timp ce fișierul **02457R01.BAT** va conține linia de comandă care va fi folosită pentru compilare, de exemplu `bpc 02457R01.PAS`.

Veți folosi `bpc` pentru compilarea fișierului sursă deoarece pentru limbajul *Pascal*, în enunțul problemei a fost specificat compilatorul inclus în mediul de programare *Borland Pascal* 7.0.

După crearea acestui fișier trebuie construită arhiva pe care o veți trimite.

Folosiți un arhivator care creează arhive `.RAR`, `.ACE` sau `.ZIP`; nu creați o arhivă autoexpandabilă.

Acum puteți trimite *e-mail*-ul care va conține soluția.

Nu uitați să atașați arhiva și să scrieți în linia *Subject*: `02457 Runda 01`.

Modificări

Redacția *GInfo* își rezervă dreptul de a modifica prezentul regulament de desfășurare, pe parcursul desfășurării concursului.

Dacă apar prevederi noi sau dacă anumite prevederi ale regulamentului sunt modificate sau eliminate, atunci acestea vor fi anunțate pe *site*-ul concursului.

Orice modificare a regulamentului va intra în vigoare, în funcție de situație, fie din momentul publicării ei, fie după încheierea rundelor aflate în desfășurare. În momentul anunțării modificării va fi precizat și momentul din care aceasta intră în vigoare.

Întrerupere și anulare

Editura *Agora Media* și redacția *GInfo* își rezervă dreptul de a întrerupe desfășurarea concursului sau de a anula întregul concurs în orice moment dacă este cazul.

Heap-uri de INTERVALE

Mihai Scorțaru

Principalul dezavantaj al heap-urilor îl constituie imposibilitatea de a determina rapid atât elementul minim, cât și elementul maxim al unei secvențe de elemente între care se poate stabili o relație de ordine. În cadrul acestui articol vom propune o structură de date care rezolvă acest dezavantaj.

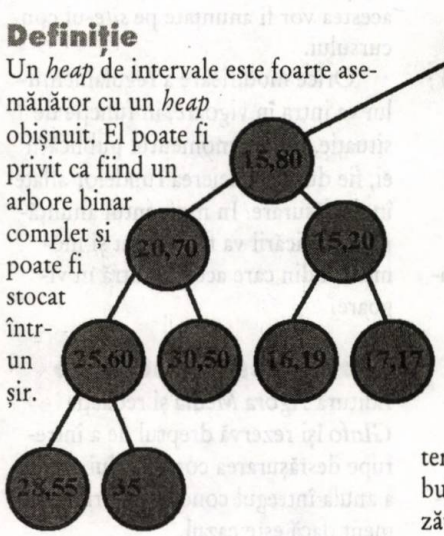
Heap-urile de intervale sunt structuri de date care permit determinarea în timp constant atât a minimului, cât și a maximului unei secvențe de elemente.

Fiind o structură de date derivată din *heap*, *heap*-ul de intervale va trebui să respecte o proprietate de *heap*, evchivalentă cu cea a *heap*-urilor obișnuite.

Pe lângă această proprietate, structura de date va trebui să permită determinarea în timp constant a elementelor minim și maxim, precum și ștergerea acestor elemente (cu păstrarea proprietății de *heap*) în timp logaritmic. Pe lângă aceste operații, *heap*-urile de intervale trebuie să permită adăugarea unui element în timp logaritmic.

Definiție

Un *heap* de intervale este foarte asemănător cu un *heap* obișnuit. El poate fi privit ca fiind un arbore binar complet și poate fi stocat într-un șir.



Totuși, există două diferențe semnificative. Prima dintre ele o constituie faptul că fiecare element al *heap*-ului va conține două valori în loc de una. Cele două valori (prima este mai mică sau egală cu a doua) reprezintă un interval. Ultimul nod, poate să conțină un singur element, în cazul în care numărul elementelor care trebuie păstrate este impar.

Ca urmare, un *heap* de intervale care trebuie să păstreze N valori, va conține un număr de elemente egal cu partea întreagă superioară a valorii $N / 2$.

Cea de-a doua diferență semnificativă este dată de proprietatea de *heap*. Pentru *heap*-urile de intervale, această proprietate specifică faptul că intervalul corespunzător unui nod trebuie să fie inclus în intervalul corespunzător părintelui nodului respectiv.

O altă formulare ar putea fi: intervalul corespunzător unui nod trebuie să includă intervalele corespunzătoare fiilor nodului respectiv.

Studiind această structură putem trage concluzia că extremitățile stângi ale intervalelor formează un *heap* obișnuit în care rădăcina conține minimul, în timp ce extremitățile drepte formează un alt *heap* obișnuit în care rădăcina conține maximul.

În imaginea de pe această pagină este prezentat un *heap* de intervale.

Observăm că, la fel ca și în cazul *heap*-urilor obișnuite, pot exista mai multe elemente cu valori egale.

Minim și maxim

Se observă imediat, pe baza proprietății de *heap* de intervale, că intervalele corespunzătoare oricărui nod sunt incluse în intervalul corespunzător rădăcinii.

Așadar, putem trage imediat concluzia că elementul minim este extremitatea stângă a intervalului corespunzător rădăcinii, iar elementul maxim este extremitatea dreaptă a acestui interval.

Ca urmare, operațiile de determinare a minimului și maximului pot fi implementate în timp constant deoarece implică doar accesarea rădăcinii arborelui (primul element al vectorului în care este stocat *heap*-ul de intervale).

O versiune a algoritmului de determinare a minimului, care prevede toate situațiile care pot apărea, este următoarea:

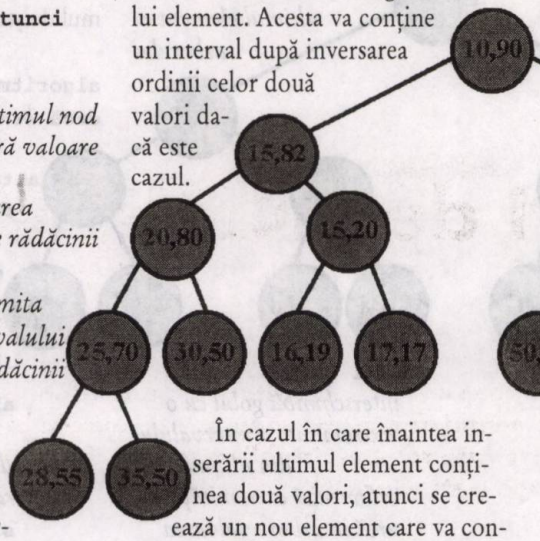
```
algoritm determinăMinim
  dacă heap-ul este vid atunci
    semnalează eroare
  altfel
    dacă rădăcina este ultimul nod
        și conține o singură valoare
      atunci
        returnează valoarea
          corespunzătoare rădăcinii
    altfel
      returnează extremita
        tea stângă a intervalului
        corespunzător rădăcinii
    sfârșit dacă
  sfârșit dacă
sfârșit algoritm
```

Algoritmul pentru determinarea maximului este foarte asemănător:

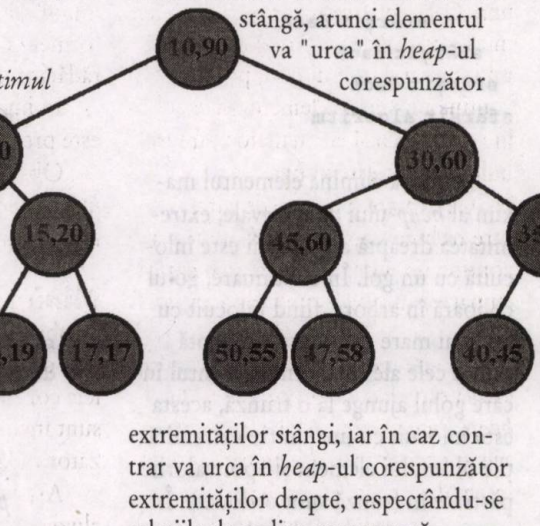
```
algoritm determinăMaxim
  dacă heap-ul este vid atunci
    semnalează eroare
  altfel
    dacă rădăcina este ultimul
      nod și conține o
      singură valoare
    atunci
      returnează valoarea
        corespunzătoare rădăcinii
    altfel
      returnează extremitatea
        dreaptă a intervalului
        corespunzător rădăcinii
    sfârșit dacă
  sfârșit dacă
sfârșit algoritm
```

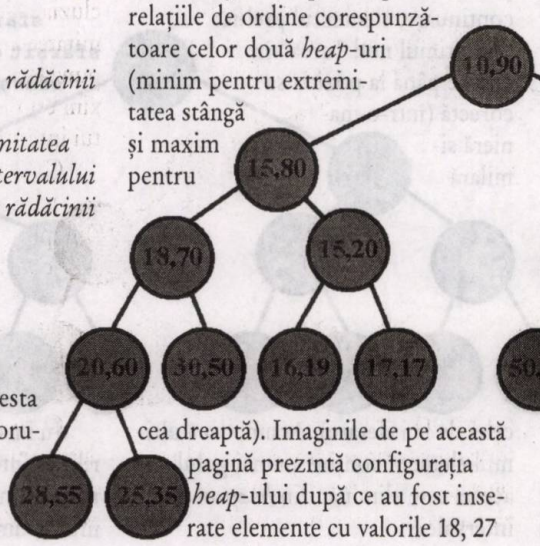
Inserarea

Pentru a insera o valoare, acesta este adăugat la sfârșitul vectorului. În cazul în care înaintea inserării, ultimul element conținea o singură valoare, atunci valoarea este adăugată ultimului element. Acesta va conține un interval după inversarea ordinii celor două valori dacă este cazul.



În cazul în care înaintea inserării ultimul element conținea două valori, atunci se creează un nou element care va conține o singură valoare. După adăugarea valorii, aceasta trebuie să "urce" în arbore într-un mod asemănător celui de la *heap*-urile obișnuite. Dacă noul element a devenit o extremitate stângă, atunci elementul va "urca" în *heap*-ul corespunzător extremităților stângi, iar în caz contrar va urca în *heap*-ul corespunzător extremităților drepte, respectându-se relațiile de ordine corespunzătoare celor două *heap*-uri (minim pentru extremitatea stângă și maxim pentru cea dreaptă). Imaginile de pe această pagină prezintă configurația *heap*-ului după ce au fost inserate elemente cu valorile 18, 27 și 82 (nu sunt inserări succesive, ci inserări pornind de la *heap*-ul prezentat anterior). Prezentăm în continuare algoritmul care realizează inserarea unui element.

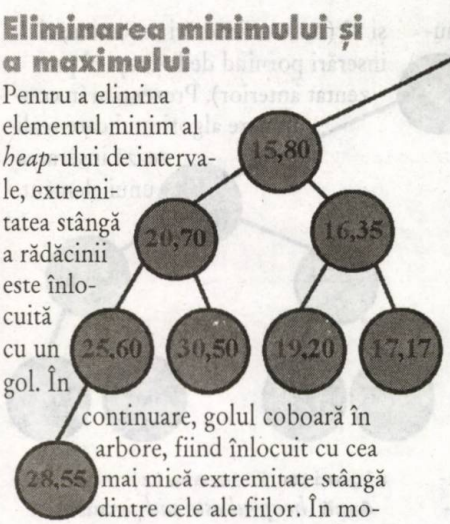





```
algoritm inserare(x):
  dacă heap-ul este vid atunci
    creează un nou nod care
      conține doar valoarea x și
      adaugă acest nod heap-ului
      ca ultim element
  altfel
    dacă ultimul element al
        heap-ului conține o
        singură valoare y
      atunci
        inserează valoarea în
          ultimul element al heap-ului
        dacă x < y atunci
          intervalul corespunzător ul-
            timului element este [x,y]
        altfel
          intervalul corespunzător ul-
            timului element este [y,x]
        sfârșit dacă
      sfârșit dacă
      cât timp x nu este în
        intervalul [a,b] corespunzător
        părintelui și x nu face parte din
        elementul rădăcină execută
        dacă x < a atunci
          interschimbă x cu a
        altfel
          interschimbă x cu b
        sfârșit dacă
      sfârșit cât timp
    sfârșit dacă
  sfârșit algoritm
```

Eliminarea minimului și a maximului

Pentru a elimina elementul minim al *heap*-ului de intervale, extremitatea stângă a rădăcinii este înlocuită cu un gol. În continuare, golul coboară în arbore, fiind înlocuit cu cea mai mică extremitate stângă dintre cele ale fiilor. În momentul în care golul ajunge la o frunză, acesta este înlocuit cu una dintre extremitățile ultimului element și apoi valorile păstrate în frunza respectivă sunt ordonate.

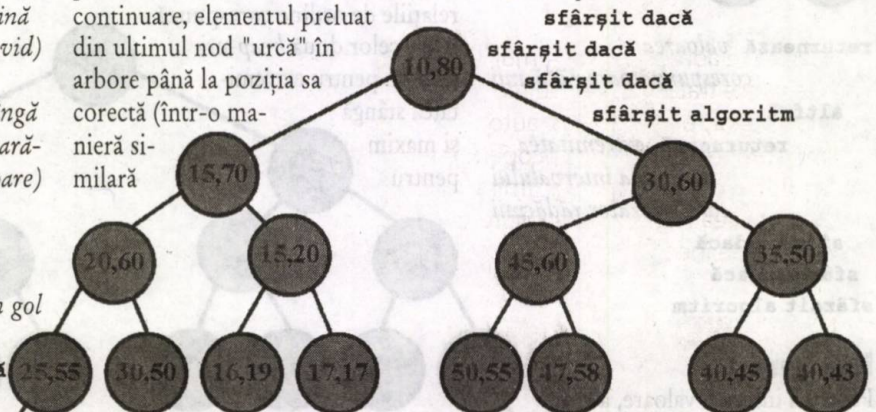


În continuare, elementul preluat din ultimul nod "urcă" în arbore până la poziția sa corectă (într-o manieră similară celei de la inserare). Acum, dacă ultimul element conține numai golul, acesta este eliminat din *heap*-ul de intervale.

Prezentăm în continuare algoritmul de ștergere a elementului minim.

```
algoritm eliminăMinim
  dacă heap-ul este vid atunci
      semnalează eroare
  altfel
      dacă rădăcina este ultimul nod
      atunci
          dacă rădăcina conține o
                singură valoare atunci
              șterge valoarea și elimină
              nodul (heap-ul devine vid)
          altfel
              șterge extremitatea stângă
                  a rădăcinii (rădăcinarămâne cu o singură valoare)
      sfârșit dacă
      altfel
          înlocuiește extremitatea
              stângă a rădăcinii cu un gol
          cât timp nodul curent
              nu este frunză execută
              interschimbă golul cu
              cea mai mică extremi-
                  tate stângă dintre
                      cele ale fiilor
          sfârșit cât timp
          dacă frunza nu este
                  ultimul element al
                      arborelui
          atunci
              interschimbă golul cu o
                  extremitate a intervalului
                      din ultimul nod
              ordonează extremitățile
              urcă în arbore valoarea
              preluată din ultimul element
          sfârșit dacă
          dacă ultimul element al
                  arborelui este vid atunci
              elimină ultimul element
          sfârșit dacă
      sfârșit dacă
  sfârșit dacă
sfârșit algoritm
```

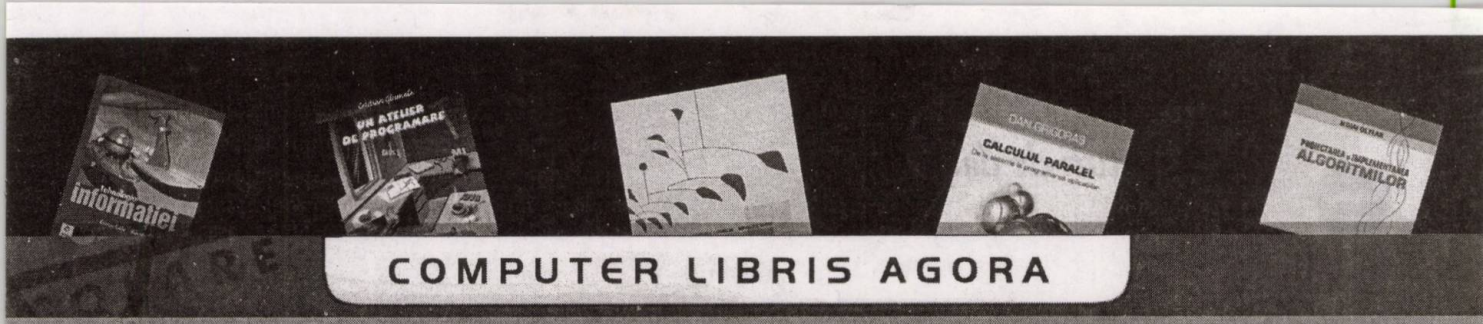
Pentru a elimina elementul maxim al *heap*-ului de intervale, extremitatea dreaptă a rădăcinii este înlocuită cu un gol. În continuare, golul coboară în arbore, fiind înlocuit cu cea mai mare extremitate dreaptă dintre cele ale fiilor. În momentul în care golul ajunge la o frunză, acesta este înlocuit cu una dintre extremitățile ultimului element și apoi valorile păstrate în frunză sunt ordonate. În continuare, elementul preluat din ultimul nod "urcă" în arbore până la poziția sa corectă (într-o manieră similară celei de la inserare). Acum, dacă ultimul element conține numai golul, acesta este eliminat din *heap*-ul de intervale.



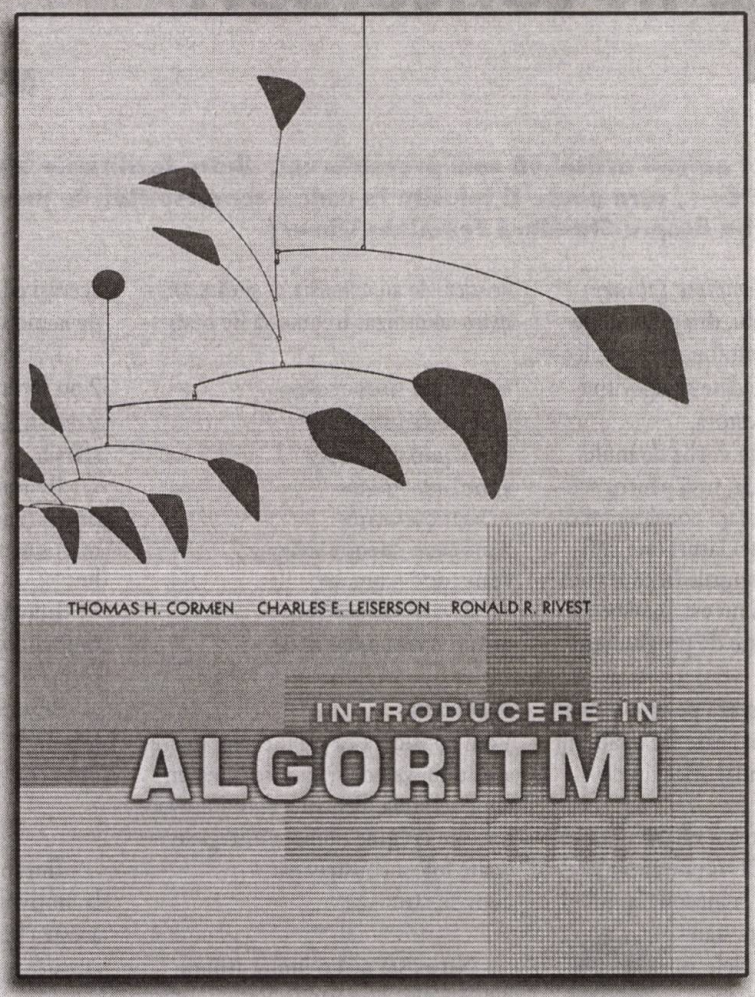
Prezentăm în continuare algoritmul de ștergere a elementului maxim.

```
algoritm eliminăMaxim
  dacă heap-ul este vid atunci
      semnalează eroare
  altfel
      dacă rădăcina este
              ultimul nod atunci
          dacă rădăcina conține o
                singură valoare atunci
              șterge valoarea și elimină
              nodul (heap-ul devine vid)
          altfel
              șterge extremitatea
              dreaptă a rădăcinii (rădăcina
              rămâne cu o singură valoare)
          sfârșit dacă
      altfel
          înlocuiește extremitatea
              dreaptă a rădăcinii cu un gol
          cât timp nodul curent nu
                este frunză execută
              interschimbă golul cu cea
                  mai mare extremitate
                  dreaptă dintre cele ale fiilor
          sfârșit cât timp
          dacă frunza nu este ultimul
              element al arborelui atunci
              interschimbă golul cu o
                  extremitate a intervalului
                      din ultimul nod
              ordonează extremitățile
          sfârșit dacă
          dacă ultimul element al
                  arborelui este vid atunci
              elimină ultimul element
              urcă în arbore valoarea
              preluată din ultimul element
          sfârșit dacă
      sfârșit dacă
  sfârșit dacă
sfârșit algoritm
```

În imagini sunt prezentate *heap*-urile de intervale obținute după eliminarea minimului, respectiv a maximului, din *heap*-ul considerat inițial.



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

STL în concursuri

Marius Andrei

În cadrul acestui articol vă vom prezenta una dintre facilitățile oferite de limbajul C++, care poate fi folosită în cadrul concursurilor de programare; este vorba despre Standard Template Library.

STL (*Standard Template Library*) privit foarte simplu, din punctul de vedere al concurentului, este o colecție de structuri de date și algoritmi aplicați asupra acestora.

Evident că este vorba de mult mai mult decât atât, însă pentru scopul articolului de față considerăm că evidențierea acestei laturi este suficientă. Pentru scopurile în care vom folosi noi *STL*-ul nu este nevoie de cunoștințe avansate de programare orientată obiect.

STL-ul se bazează pe câteva *template*-uri (șabloane) care fac aplicabili toți algoritmii pentru orice tip de structuri de date.

Pentru a vă arăta importanța *STL*-ului trebuie precizat de la început faptul că această librărie este standard. Toate exemplele au fost testate atât cu *gcc*, cât și cu *Visual C++*.

Așadar, librăria poate fi folosită în concursuri, atât în cele pentru liceu (olimpiade), cât și în cele studențești (*ACM*). Datorită faptului că librăria este relativ nouă, ea nu este disponibilă în *Borland C++* 3.1.

Prima informație de care avem nevoie pentru a putea utiliza *STL* sunt fișierele ce trebuie incluse. Includem mai mult decât ne este necesar, însă aceste *header*-e ne vor folosi mai încolo, în exemplele următoare.

Veți observa că în partea de început a programului nu apare nimic anormal, în afară de faptul că nu toate *header*-ele au extensia *.h* și că *namespace*-ul utilizat trebuie să fie `std`:

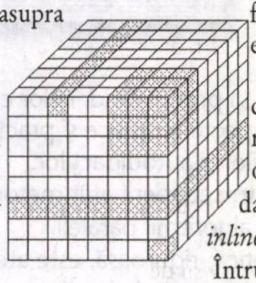
```
#include <iostream>
#include <string>
#include <vector>
#include <set>
#include <map>
#include <algorithm>
#include <queue>
#include <assert.h>
using namespace std;
```

Vectori

Vom considera ca prim exemplu crearea unui vector și sortarea elementelor acestuia. Pentru a defini un vector care conține numere întregi se poate folosi construcția:

```
vector<int> a;
```

Nu există nici o limită asupra dimensiunii vectorului. Ea poate fi precizată de la început, însă nu este obligatoriu. De asemenea, putem preciza și o valoare inițială pentru elementele vectorului respectiv:



```
vector<int> a(50,1);
```

În exemplul anterior este definit un vector `a` care conține `50` de elemente care sunt inițializate cu valoarea `1`.

Accesarea unui element se realizează, așa cum suntem obișnuiți, folosind construcția `a[i]`, pentru al `i`-lea element, numerotarea fiind aceeași cu cea din *C/C++*, (începând de la indicele `0`).

Dacă vrem să adăugăm elementul 2 nu trebuie decât să scriem `a.push_back(2)`, și l-am adăugat la sfârșit.

Redimensionarea se realizează prin `a.resize(15)`. Astfel putem mări sau micșora lungimea vectorului.

Lungimea vectorului ne este tot timpul disponibilă printr-un apel de forma `a.size()`.

Sortarea vectorului se realizează folosind construcția:

```
sort(&a[0],&a[N]); //N reprezintă lungimea vectorului
```

Întrebarea evidentă aici se referă la motivul pentru care cineva ar folosi aceste clase, când poate face totul foarte simplu, folosind, de exemplu, `qsort`.

Primul lucru spectaculos este că sortarea rulează de până la 20 de ori mai repede, în special datorită implementărilor *inline*.

Întrucât uneori ne permitem de 20 de ori mai mult timp pentru o sortare, s-ar putea spune că nu se câștigă prea mult, dar mai e mult de învățat de la *STL*.

Funcția `sort` are doi parametri: primul element și elementul imediat următor ultimului element.

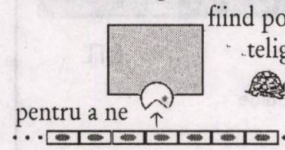
De obicei se scrie:

```
sort(a.begin(), a.end()),
```

însă pentru a înțelege această construcție trebuie să știm câte ceva despre iteratori.

Iteratori

Trebuie să ne gândim la iteratori ca fiind pointeri "inteligenți", folosiți pentru a ne deplasa printre elementele structurii noastre de date (pentru toate tipurile de structuri, nu doar pentru vectori).



De exemplu, declararea unui iterator pe o structură de tip vector de numere întregi se poate face cu ajutorul construcției:

```
vector<int>::interator i;.
```

Pentru aceste structuri de date există doi iteratori impliciți și anume `begin()` și `end()`, care permit obținerea unui iterator pentru primul element, respectiv pentru elementul de după ultimul element.

Pe parcurs vom prezenta mai multe detalii referitoare la folosirea iteratorilor.

Cozi de priorități

Declararea unei cozi de priorități (*heap*) se realizează cu ajutorul unei construcții de forma:

```
priority_queue<int> a;
```

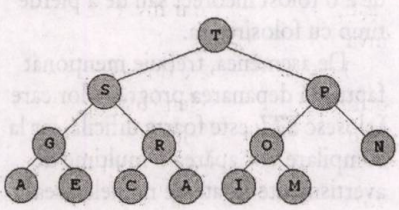
De obicei implementarea unei cozi de priorități se face cu ajutorul *heap*-urilor.

Introducerea de elemente într-o coadă de priorități se realizează folosind o construcție similară cu:

```
a.push(8);
```

Determinarea dimensiunii cozii de priorități se face cu ajutorul funcției `size()` specifică structurii de date:

```
int dim = a.size();
```



Determinarea valorii elementului cu cea mai mare prioritate (elementul din vârful *heap*-ului) se realizează cu ajutorul funcției `top()` specifică structurii de date:

```
int varf = a.top();
```

Extragerea elementului cu cea mai mare prioritate se realizează cu ajutorul funcției `pop()` specifică structurii de date:

```
a.pop();
```

Există susținători ai ideii potrivit căreia implementarea unui *heap* este imediată și este preferabil să avem control total asupra structurii de date pe care o implementăm.

Există argumente în favoarea acestei idei dar, din alt punct de vedere, ar putea fi de preferat să știm sigur că nu vom greși și că vom mai câștiga câteva minute care se pot dovedi prețioase.

Tabele de dispersie

Cu ajutorul acestor tabele se reprezintă o relație 1 la 1 între două tipuri de date. Primul element este cheia, iar al doilea este valoarea. În exemplul următor vom asocia câte un număr întreg fiecărui oraș:

```
map<string,int> a;
a["Bucuresti"] = 10;
a["Cluj"] = 7;
a["Iasi"] = 3;
a["Constanta"] = 8;
```

Dacă dorim să tipărim numărul corespunzător *Clujului* vom scrie:

```
cout << a["Cluj"]
```

Efectul acestei construcții va consta în afișarea valorii 7 la ieșirea standard.

Dacă nu există cheia, se va afișa valoarea 0 (în general, se va lua în considerare constructorul implicit dacă este vorba de alt tip de dată decât număr întreg).

Astfel, `a["Timisoara"]` va avea valoarea 0.

Este interesant de menționat faptul că structura de date creată prin `map<int, int>` poate fi privită ca un vector de întregi infinit de lung (la ambele capete).

Deja începem să ne dăm seama de câteva dintre avantajele *STL*-ului.

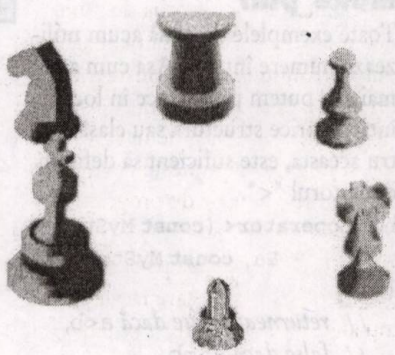
Mulțimi

Poate cel mai folositor lucru pentru un concurent sunt mulțimile. Ele se declară folosind o construcție de tipul:

```
set<int> a;
```

Operațiile de bază cu mulțimile sunt inserarea, ștergerea și căutarea de elemente care se pot realiza cu ajutorul funcțiilor `insert()`, `erase()` și `find()`:

```
a.insert(7);
a.erase(5);
if (a.find(7) != a.end())
  cout << Gasit;
```



Pe lângă aceste operații de bază, sunt definite intersecția, reuniunea și diferența.

Implementarea operațiilor cu mulțimi nu mai pare a fi foarte simplă, iar una cu arbori echilibrați în mod sigur nu e facilă, mai ales în momentul în care trebuie implementat algorimtul de ștergere a unui nod.

În plus, orice operație are ordinul de complexitate $O(\log N)$!

Pe lângă timpul câștigat (puțini se pot lăuda cu numai 20 de minute pentru a implementa un arbore *AVL* cu ștergeri), nu trebuie să ne facem griji că ar putea exista greșeli de implementare.

Folosirea iteratorilor

Pentru exemplificare vom parcurge elementele unei mulțimi și elementele unei tabele de dispersie.

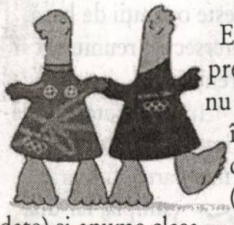
```
set<int> a;
map<string,string> b;

for(set<int>::iterator i =
   a.begin(); i != a.end(); ++i)
  cout << *i << endl;
for(map<string,string>::
        iterator i = b.begin();
              i != b.end(); ++i)
   cout << i->first << " => " <<
             i->second << endl;
```

Este necesar să folosim construcția "`!= a.end()`" și nu "`< a.end()`", deoarece datele nu sunt obligatoriu ordonate în memorie.

make_pair

Toate exemplele de până acum utilizează numere întregi. Așa cum am mai zis, putem pune orice în loc de întregi, orice structură sau clasă. Pentru aceasta, este suficient să definim operatorul "<".

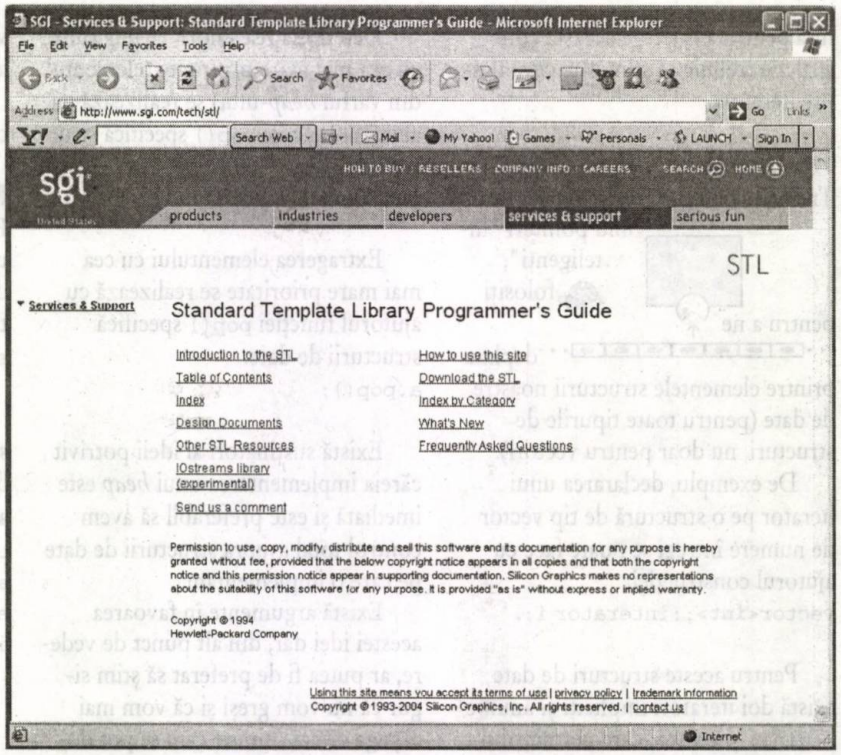
```
bool operator< (const MyStruct
          &a, const MyStruct &b)
{
  // returnează true dacă a<b,
  // false dacă a>=b
}
```



Există o clasă predefinită care nu face decât să încapsuleze două date (oricare două date) și anume clasa `pair`.

Ca exemplu, vom prezenta o secvență în care vom declara un vector ale cărui elemente sunt perechi de numere întregi și vom sorta elementele vectorului respectiv.

```
int N,x,y;
vector< pair<int,int> > a;
//este esențial să existe un spațiu
//între ">" și ">"!
cin >> N;
for(int i=0;i<N;++i) {
  cin >> x >> y;
  a.push_back(make_pair(x,y));
  // make_pair creează o clasă
  // pair<int,int>
}
sort(a.begin(),a.end());
//sortează după prima coordonată
//(primul întreg),
//iar în caz de egalitate după a doua
//coordonată.
```



Mult mai mult

Tot ce am prezentat este doar o mică parte din facilitățile oferite de *STL*.

Pentru o prezentare completă vă recomandăm să accesați adresa **http://www.sgi.com/tech/stl/**.

Este clar că folosirea *STL*-ului reprezintă un avantaj. Timpul de implementare este redus, timpul de rulare este mai mic, posibilatea apariției erorilor în structura noastră de date este inexistentă.

Unii concurenți ar putea să protesteze împotriva folosirii *STL*-ului. Vor spune că nici nu mai trebuie să știi arbori *AVL* (sau arbori roșu-negru), nici *hash*-uri, nici măcar un *heap*. Vom deveni simplii utilizatori ai structurilor, în loc să le putem implementa de fiecare dată. Poate e adevărat, însă *STL*-ul nu ne ajută să găsim metoda de rezolvare. Ne ajută doar să implementăm mai repede rezolvarea.

Dar în concursuri, față de cei care folosesc limbajul *Pascal*, *STL* oferă un avantaj fără discuții. Deși nu mai sunt foarte mulți programatori *Pascal*, aceștia există, mulți dintre ei fiind chiar foarte buni. Aceștia vor acuza folosirea *STL*-ului și dintr-un anumit punct de vedere au dreptate. Problemele propuse sunt în general alese în așa fel încât să ofere șanse egale atât programatorilor în C++, cât și celor care programează în *Pascal*. Acum apare un avantaj în favoarea celor care aleg limbajul C++.

Totuși, nu putem interzice folosirea *STL*-ului deoarece această librărie este un standard.

STL-ul este o evoluție, însă nici nu trebuie privit ca un element care trebuie obligatoriu cunoscut.

Pentru a-l utiliza în concursuri, este necesară o cunoaștere bună a librăriei, pentru că altfel apare riscul de a o folosi incorect sau de a pierde timp cu folosirea sa.

De asemenea, trebuie menționat faptul că depanarea programelor care folosesc *STL* este foarte dificilă, iar la compilare vor apărea o mulțime de avertismente legate de numele prea

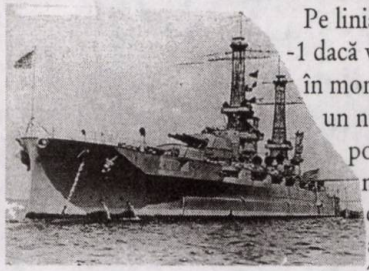
lungi ce vor fi trunchiate la 255 de caractere.

Un exemplu complet

La final vom prezenta un exemplu complet de folosire a *STL*-ului.

Problema a fost propusă spre rezolvare în tabara de pregătire de la *Orăștie*. Enunțul și rezolvarea îi aparțin lui *Radu Berinde*.

Enunț



Pe mare va avea loc o mare bătălie între N vapoare. Vapoarele sunt considerate a fi puncte și sunt date prin coordonatele lor carteziene x și y.

Din motive greu de înțeles, vapoarele nu pot ataca decât vapoarele care se află la stânga și mai jos (mai exact, un vapor la poziția x_1, y_1 poate ataca alt vapor la poziția x_2, y_2 dacă și numai dacă $x_1 > x_2$ și $y_1 > y_2$).

Deoarece această bătălie are loc în zona *Triunghiului Bermudelor*, vapoarele apar (se teleportează) pe rând în zona bătăliei. Vapoarele sunt identificate prin numere întregi cuprinse între 1 și N, în ordinea apariției lor.

În momentul în care un vas apare, dacă există alt vas care a apărut deja și care poate să îl atace pe cel nou, vasul nou este distrus instantaneu. Dacă nu, vasul cel nou rămâne pe mare și distruge toate vasele pe care le poate ataca.

Dându-se coordonatele la care apar pe rând vapoarele, să se afle pentru fiecare vapor dacă este distrus sau nu în momentul apariției sale și dacă nu este distrus, să se precizeze numărul total de vapoare rămase pe mare după apariția sa.

Date de intrare

Pe prima linie a fișierului de intrare **sea2.in** se află numărul natural N reprezentând numărul de vapoare ce vor apărea pe mare.

Pe fiecare dintre următoarele N linii se află câte o pereche de numere întregi separate printr-un spațiu, reprezentând coordonata x respectiv y a vaporului corespunzător liniei (mai exact, pe linia i din fișier sunt scrise coordonatele vaporului identificat prin i - 1, pentru orice valoare i cuprinsă între 2 și $N + 1$).

Date de ieșire

Fișierul de ieșire **sea2.out** va conține N linii, pe fiecare dintre acestea aflându-se câte un număr întreg.

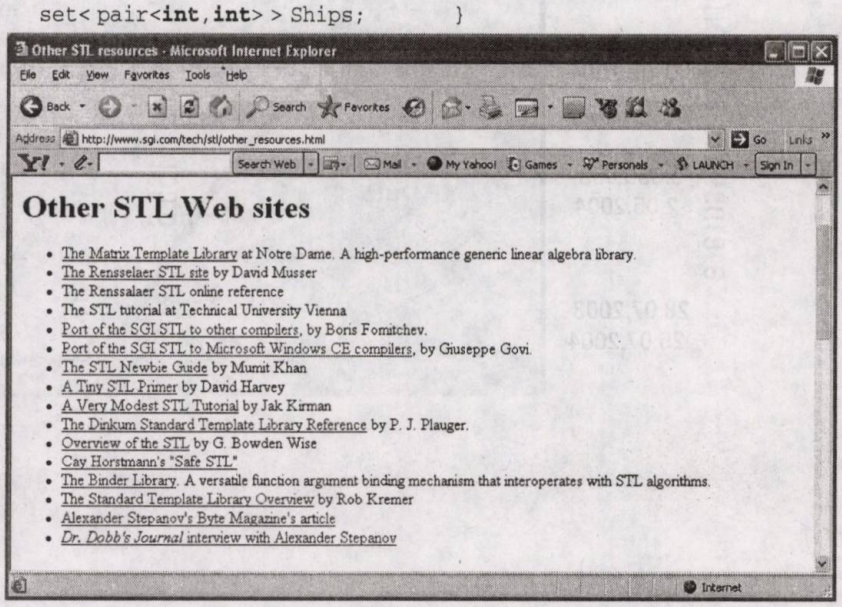
Pe linia i se va afla valoarea -1 dacă vaporul i este distrus în momentul apariției, sau un număr întreg strict pozitiv reprezentând numărul de vapoare de pe mare după apariția vaporului i în caz contrar.

Restricții și precizări

- $1 \le N \le 200.000$;
- coordonatele sunt numere întregi strict pozitive mai mici sau egale cu 260000;
- nu vor exista două vase cu aceeași coordonată x sau cu aceeași coordonată y.

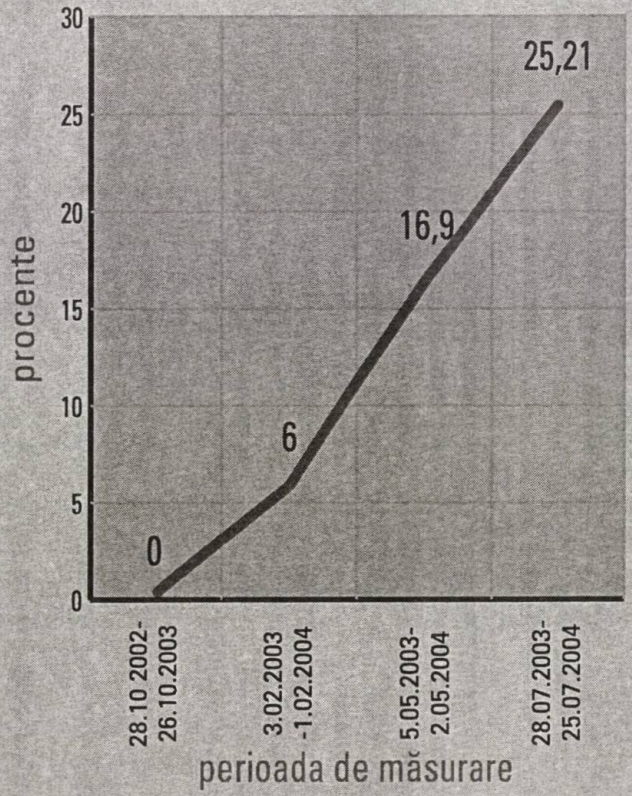
Soluție

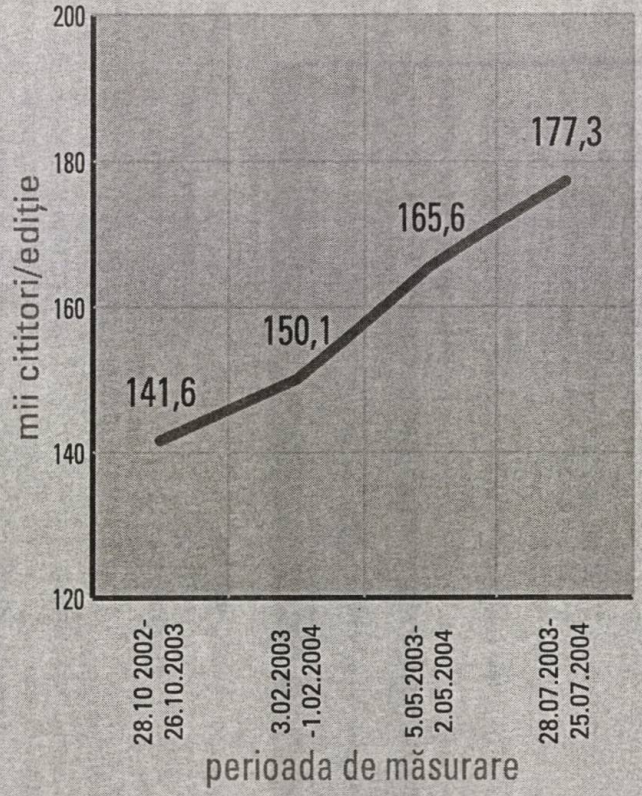
```
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <set>
#include <utility>
using namespace std;

set< pair<int,int> > Ships;
set< pair<int,int> >::iterator
                        It, It2, t;

int main() {
  FILE *fi = fopen("sea2.in",
                          "rt");
  FILE *fo = fopen("sea2.out",
                          "wt");
  int x, y, nr;

  for (fscanf(fi, "%d", &nr);
                      nr; nr--) {
    fscanf(fi, "%d %d", &x, &y);
    It = Ships.upper_bound(
              make_pair(x, y));
    if (It != Ships.end() &&
              It->second > y) {
      fprintf(fo, "-1\n");
      continue;
    }
    if (Ships.size()) {
      for (It2 = It;
       It2 != Ships.begin() &&
              (--(t = It2))->
                    second < y;
       It2 = t
      );
      Ships.erase(It2, It);
    }
    Ships.insert(
              make_pair(x, y));
    fprintf(fo, "%d\n",
                Ships.size());
  }
  fclose(fo);
  fclose(fi);
  return 0;
}
```



PC MAGAZINE ROMÂNIA

Numărul cititorilor PC Magazine a crescut cu 25 %!





Sursa:
Studiul Național de Audiență. Perioada de măsurare: 28 iulie 2003 – 25 iulie 2004.
Studiul a fost realizat pe un eșantion de 14.857 persoane, reprezentativ pentru populația României din centrele urbane cu mai mult de 50.000 locuitori și vârsta între 14 și 64 ani.
Eșantionul utilizat pentru această livrare de date conține un număr de 11.160 interviuri comune cu cel utilizat pentru livrarea de date anterioară
(interviurile realizate în perioada 28 iulie 2003 – 2 mai 2004).
Marja de eroare a Studiului Național de Audiență este de 0,804%.

Modele de PROIECTARE

Tudor Orha

Modelele de proiectare constituie un element foarte important în proiectarea aplicațiilor software. În această serie de articole vom prezenta câteva modele de design utile în dezvoltarea sistemelor informatice.

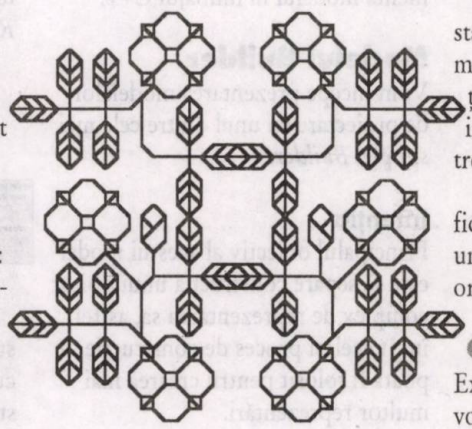
Ce este un model de design?

Orice model descrie o problemă ce apare în mod repetat în mediul nostru de lucru, după care descrie principiul soluției respectivei probleme, într-un asemenea fel încât să poată fi refolosit de milioane de ori, scrie *Cristopher Alexander* în *A Pattern Language*.

Această definiție este valabilă în orice domeniu, atât în proiectarea și construcția de clădiri sau avioane, cât și în proiectarea și construcția de *software*.

Elementele constitutive ale unui model sunt prezentate în continuare:

- *Numele* modelului reprezintă o modalitate de a descrie o problemă de design, soluțiile și consecințele lor într-un cuvânt sau două. Aceasta ajută la formarea unui vocabular comun, ceea ce conduce la o mai ușoară comunicare cu alte persoane.
- *Problema* descrie situațiile în care se poate aplica un model. Se explică problema și contextul ei. Uneori, se pot descrie probleme specifice; de exemplu se poate arăta cum pot fi reprezentați algoritmii ca obiecte. Alteori se pot descrie structuri de clase și obiecte care sunt caracteristice unui *design* inflexibil.
- *Soluția* descrie elementele care compun modelul, relațiile dintre ele, responsabilitățile și modul în care elementele colaborează. Soluția nu va descrie un model concret sau o implementare, pentru că un model este un șablon care poate fi aplicat în multe situații diferite. Modelul va furniza o descriere abstractă a unei probleme de design și cum poate fi aceasta rezolvată aranjând diferitele elemente într-un anumit mod.



- *Consecințele* sunt rezultatul și costurile aplicării unui model. Ele sunt critice în evaluarea modelelor alternative și în înțelegerea costurilor și beneficiilor aplicării modelului. De multe ori consecințele în cazul *software*-ului sunt legate de compromisuri de spațiu și timp; ele pot fi legate de limbajul în care se realizează implementarea. Datorită faptului că reutilizarea este de multe ori un factor în designul orientat pe obiecte, consecințele unui model includ impactul lui asupra flexibilității, extensibilității sau portabilității sistemului. Dezvăluirea acestor consecințe va ajuta la înțelegerea și evaluarea lor.

Un model de design numește, abstractizează și identifică aspectele cheie ale unei structuri de modelare care îl face util în crearea unui model orientat obiect reutilizabil.

Modelul identifică clasele și instanțele participante, rolul acestora și modul în care colaborează, distribuția responsabilităților între clase și instanțe. Fiecare model se concentrează pe o problemă particulară.

Ca limbaj în care vom exemplifica modelele am ales C++, ca fiind unul dintre cele mai populare limbaje orientate pe obiecte.

Clasificarea modelelor

Există multe modele care au fost dezvoltate de-a lungul timpului și doar câteva dintre ele vor fi prezentate în acest serial. Acestea sunt clasificate în funcție de scopul lor.

Pe baza acestui criteriu, avem modele:

- *creaționale*: descriu procesul creării obiectelor;
- *structurale*: descriu compoziția obiectelor;
- *comportamentale*: descriu modul în care sunt împărțite responsabilitățile între obiecte.

Modul de prezentare a modelelor

Modelele vor fi prezentate folosind un format consistent. Fiecare model

va fi impărțit în secțiuni conform șablonului următor. Astfel vom avea o prezentare uniformă a informației care va face modelele mai ușor de învățat, comparat și utilizat.

Numele și clasificarea

Numele modelului exprimă succint esența modelului. Un nume bun este esențial deoarece va intra în vocabularul proiectantului. Clasificarea reflectă schema de clasificare a modelelor introdusă anterior. Numele va fi prezentat în limba engleză.

Intenția

Este o afirmație scurtă care răspunde următoarelor întrebări:

- *Ce face modelul?*
- *Care este rațiunea și scopul lui?*
- *Ce problemă de design adresează?*

Alte denumiri

Majoritatea modelelor sunt cunoscute și sub alte denumiri decât cea utilizată cel mai frecvent.

Motivație

Este prezentat un scenariu care ilustrează o problemă de *design* și cum sunt utilizate structurile de clase și obiecte din model pentru a rezolva problema. Acest scenariu va fi util în înțelegerea descrierii mai abstracte care va urma.

Aplicabilitate

Sunt descrise situațiile în care poate fi aplicat modelul; de asemenea, în această secțiune se pot exemplifica *design*-uri neperformante. Scopul acestei secțiuni este descrierea modalității în care pot fi recunoscute aceste situații.

Structura

Structura conține o reprezentare grafică a claselor, colaborărilor și interacțiunilor din model folosind o reprezentare bazată pe *UML*. Nu vom prezenta în această serie de articole această notație.

Participanți

Sunt descrise clasele și/sau obiectele care participă în model și care sunt responsabilitățile fiecărui element prezentat.

Colaborări

Este descris modul în care participanții colaborează pentru a-și îndeplini responsabilitățile.

Consecințe

Este prezentat modul în care modelul îndeplinește obiectivul său, care sunt compromisurile și rezultatele folosirii modelului și care sunt părțile din structura sistemului care vor putea varia independent.

Implementare

Sunt prezentate probleme, sugestii sau tehnici care ar trebui să fie cunoscute în momentul implementării modelului.

Exemple

Sunt prezentate fragmente de cod care ilustrează cum s-ar putea implementa modelul în limbajul C++.

Modelul Builder

Vom începe prezentarea modelelor de proiectare cu unul dintre cele mai simple: *Builder*.

Intenția

Principalul obiectiv al acestui model este să separe construcția unui obiect complex de reprezentarea sa, astfel încât același proces de construcție să poată fi folosit pentru crearea mai multor reprezentări.

Motivație

O aplicație pentru citirea documentelor în format *RTF* (*Rich Text Format*) ar putea fi capabilă să convertească documentele *RTF* în diverse alte formate.

De exemplu, ar putea să convertească documentul în text simplu (fără formatări) sau în format *PDF*. Problema este că lista de formate în care se poate face conversia este deschisă. Așadar, ar trebui să fie ușor de adăugat un nou format fără a modifica operația de citire.

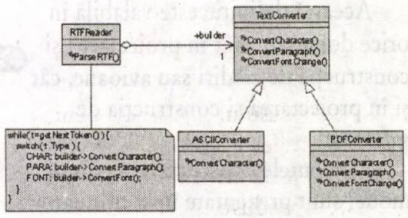
O soluție este să se configureze clasa `RTFReader` cu un obiect `TextConverter` care convertește formatul *RTF* într-o altă reprezentare a textului.

Pe măsură ce `RTFReader`-ul parcurge documentul, se folosește de `TextConverter` pentru a executa conversia.

De fiecare dată când `RTFReader` găsește un element *RTF* (fie că este text sau element de control), va emite câte o cerere către `TextConverter` ca acesta să convertească elementul.

Obiectele `TextConverter` sunt responsabile să execute conversia datelor și să reprezinte elementul în formatul corespunzător.

Subclasele clasei `TextConverter` sunt specializate pentru diferite formate și conversii. De exemplu, un `ASCIIConverter` va ignora orice cerere diferită de conversia de text, iar un `PDFConverter` va interpreta toate mesajele pentru a produce un document *PDF* care conține toate elementele stilistice prezente în documentul *RTF*.



Fiecare clasă convertor preia construcția unui obiect complex și o ascunde în spatele unei interfețe abstracte.

Convertorul (constructorul) este separat de componenta de citire, a cărei responsabilitate este de a citi și interpreta un document *RTF*.

Fiecare clasă convertor se numește *builder* în acest model, iar clasa `RTFReader` este denumită *director*.

Aplicat acestui exemplu, modelul *Builder* separă algoritmul de interpretare a unui format text de modul în care este creat și reprezentat un format convertit.

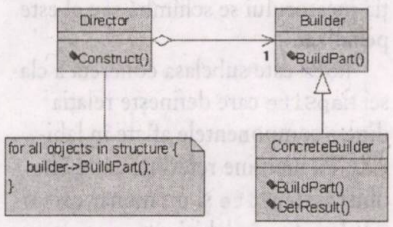
Aceasta permite refolosirea clasei `RTFReader` pentru a crea diferite reprezentări text din documente *RTF*, doar prin configurarea sa cu diferite subclase ale clasei `TextConverter`.

Aplicabilitate

Modelul *Builder* se poate folosi în următoarele situații:

- algoritmul pentru crearea unui obiect complex trebuie să fie independent de părțile care compun obiectul și felul în care ele sunt asamblate.
- procesul de construcție trebuie să permită crearea de reprezentări diferite ale obiectului care trebuie construit.

Structura



Participanți

- *Builder* (`TextConverter`):
  - ♦ definește o interfață abstractă pentru crearea părților unui obiect *Produs*;
- *ConcreteBuilder* (`ASCIIConverter`, `PDFConverter`):
  - ♦ construiește și asamblează părți ale produsului prin implementarea interfeței *Builder*;
  - ♦ definește și menține o reprezentare a produsului;
  - ♦ furnizează o interfață pentru a obține produsul (obiectul final rezultat în urma construcției);
- *Director* (`RTFReader`):
  - ♦ construiește un obiect cu ajutorul interfeței *Builder*;
- *Produsul* (text *ASCII*, document *PDF*):
  - ♦ reprezintă obiectul complex aflat în construcție. *ConcreteBuilder* construiește reprezentarea internă a produsului și definește procesul prin care este asamblat;
  - ♦ include clasele care definesc părțile constitutive, inclusiv interfețele pentru asamblarea produsului final.

Colaborări

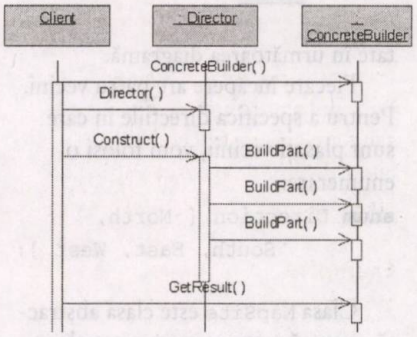
Clientul crează obiectul *Director* și îl configurează cu obiectul *Builder* dorit.

Obiectul *Director* notifică *Builder*-ul de fiecare dată când o parte a *Produsului* trebuie construită.

Builder-ul prelucrează cererile de la *Director* și adaugă părți la *Produs*.

Clientul recuperează *Produsul* de la *Builder*.

Următoarea diagramă ilustrează cum *Builder*-ul și *Director*-ul cooperează cu un client.



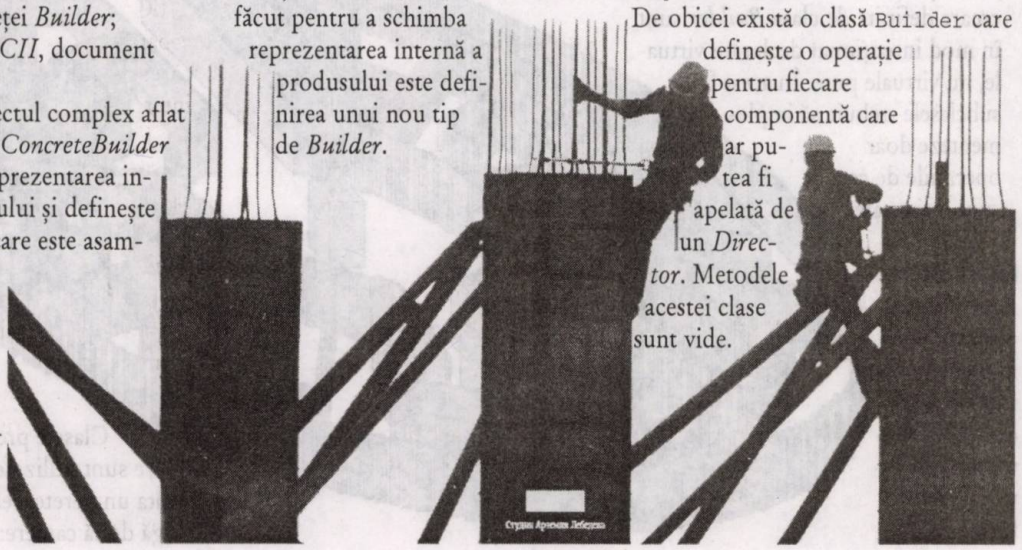
Consecințe

Principalele consecințe sunt următoarele:

- Permite modificarea reprezentării interne a produsului. Obiectul *Builder* furnizează o interfață abstractă pentru construirea produsului. Interfața permite *Builder*-ului să ascundă reprezentarea și structura internă a produsului. De asemenea, modul în care este asamblat produsul este ascuns. Deoarece produsul este construit printr-o interfață abstractă, tot ceea ce trebuie făcut pentru a schimba reprezentarea internă a produsului este definirea unui nou tip de *Builder*.
- Izolează codul pentru construcție și reprezentare. Modelul *Builder* îmbunătățește modularizarea prin încapsularea modului în care un obiect complex este construit și reprezentat. *Clienții* nu au nevoie să cunoască detalii despre clasele care definesc structura internă a produsului; astfel de clase nu apar în interfața *Builder*-ului. Fiecare *ConcreteBuilder* conține tot codul necesar creării și asamblării unui tip de produs. Codul este scris o singură dată, după care diferiți *Directori* pot să-l reutilizeze pentru a construi *Produse* din alte surse. Pentru exemplul anterior, am putea defini un cititor dintr-un format diferit de *RTF*, de exemplu *HTML*, și să utilizeze clasele `ASCIIConverter` sau `PDFConverter` pentru a extrage textul sau a crea documente *PDF*.
- Asigură un control fin asupra procesului de construcție. Spre deosebire de modelele care construiesc produsele într-o singură etapă, modelul *Builder* construiește produsul pas cu pas, sub controlul *Directorului*. Doar când produsul este terminat *Directorul* îl va obține de la *Builder*. Ca urmare, interfața *Builder* reflectă procesul de construcție a *Produsului* mai mult decât alte modele. Aceasta conferă mai mult control asupra procesului de construcție și ca urmare și asupra structurii interne a obiectului rezultat.

Implementare

De obicei există o clasă `Builder` care definește o operație pentru fiecare componentă care ar putea fi apelată de un *Director*. Metodele acestei clase sunt vide.



Clasele care sunt derivate din ea vor implementa doar operațiile pentru componentele de care sunt interesate.

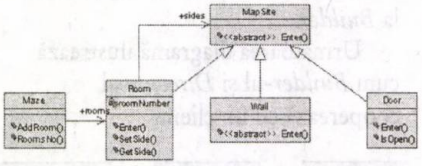
Alte probleme de implementare care se pot considera sunt:

- *Construcția și interfața pentru construirea Produsului.* Clasa `Builder` construiește obiectele pas cu pas. Ca urmare, interfața clasei `Builder` trebuie să fie suficient de generală pentru a permite construcția de produse pentru toate felurile de *Builder*-e. Lucrurile sunt simple atunci când construcția obiectului constă doar din adăugarea de elemente la sfârșitul unei colecții. Mai complicată este situația în cazul în care este necesar accesul la părți ale produsului, de exemplu când se construiesc arbori pornind de la bază (*bottom-up*). În acest caz, *Builder*-ul va returna noduri fii *Directorului*, care la rândul său le va transmite înapoi *Builder*-ului pentru a construi nodurile părinte.
- *Clasă abstractă pentru Produse.* În general, *Produsele* create de către *Builder*-e diferă suficient de mult în reprezentarea lor încât e prea puțin de câștigat prin generalizarea *Produselor*. Deoarece de obicei clientul este cel care configurează *Directorul* cu obiectul *Builder*, clientul este în măsură să recupereze produsul *Builder*-ului conform cu interfața acestuia.
- *Metode vide în implementarea implicită pentru Builder.* Metodele de creare definite de clasa *Builder* sunt în mod intenționat declarate virtuale, nu virtuale pure. În acest fel, subclasele trebuie să implementeze doar operațiile de care sunt interesate.

Exemplu

Pentru exemplificare vom implementa o funcție `CreateMaze` pentru a construi un labirint format din încăperi care va fi folosit pentru un joc. Fiecare încăpere conține detalii despre vecinii săi; vecinii pot fi o altă încăpere, un perete sau o ușă către o altă încăpere.

Clasele `Room`, `Door` și `Wall` definesc componentele labirintului. Relațiile între aceste clase sunt prezentate în următoarea diagramă.



Fiecare încăpere are patru vecini. Pentru a specifica direcțiile în care sunt plasați vecinii, vom folosi o enumerare:

```
enum Direction { North,
                 South, East, West };
```

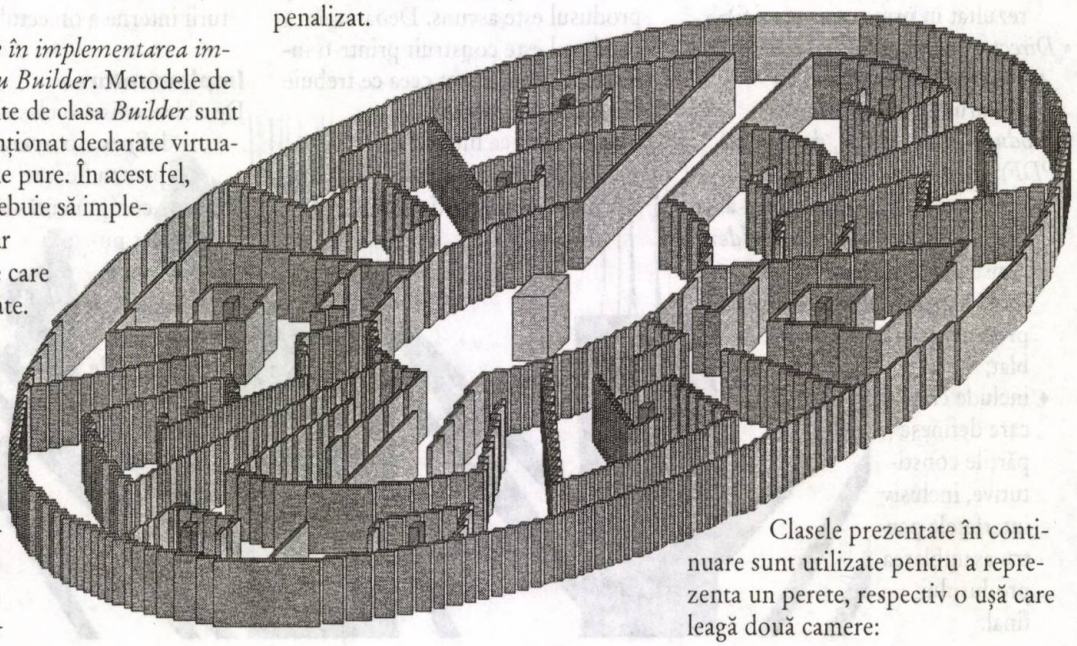
Clasa `MapSite` este clasa abstractă comună tuturor componentelor labirintului. Pentru a simplifica exemplul, `MapSite` definește o singură metodă: `Enter`. Semnificația ei depinde de locul în care se intră. Dacă se intră într-o cameră, atunci locația jucătorului se schimbă. Dacă jucătorul intră într-o ușă, unul din următoarele două evenimente poate avea loc: dacă ușa este deschisă, atunci el va intra în camera următoare; dacă ușa este închisă, atunci jucătorul va fi penalizat.

```
class MapSite {
public:
  virtual void Enter() = 0;
};
```

Metoda `Enter` furnizează o bază simplă pentru operațiile complexe ale jocului. De exemplu, dacă jucătorul se află într-o încăpere și decide să se deplaseze spre est, jocul poate să determine ce obiect `MapSite` se află în imediata vecinătate spre est și să apeleze metoda `Enter` a acelui obiect. Această operație va decide dacă locația jucatorului se schimbă sau el este penalizat.

`Room` este subclasa concretă a clasei `MapSite` care definește relația dintre componentele aflate în labirint. Ea menține referințe către alte obiecte `MapSite` și un număr care o va identifica în labirint.

```
class Room : public MapSite {
public:
  Room(int roomNo);

  MapSite* GetSide(Direction)
                          const;
  void SetSide(Direction,
                    MapSite*);
  virtual void Enter();
private:
  MapSite* sides[4];
  int roomNumber;
};
```



Clasele prezentate în continuare sunt utilizate pentru a reprezenta un perete, respectiv o ușă care leagă două camere:

```
class Wall : public MapSite {
public:
  Wall();

  virtual void Enter();
};

class Door : public MapSite {
public:
  Door(Room* = 0, Room* = 0);
  virtual void Enter();
  Room* OtherSideFrom(Room*);

private:
  Room* room1;
  Room* room2;
  bool isOpen;
};
```

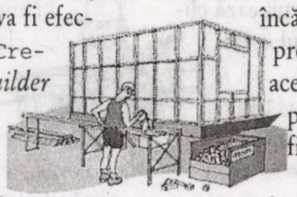
Mai multe informații despre labirint sunt păstrate în clasa `Maze`. Aceasta va conține și o colecție de camere. `Maze` poate să găsească o încăpere pe baza numărului ei de ordine folosind metoda `RoomNo`.

```
class Maze {
public:
  Maze();
  void AddRoom(Room*);
  Room* RoomNo(int) const;

private:
  // ...
};
```

O implementare a funcției `CreateMaze` care ignoră modelele de design ar putea fi următoarea:

```
Maze* CreateMaze () {
  Maze* aMaze = new Maze;
  Room* r1 = new Room(1);
  Room* r2 = new Room(2);
  Door* theDoor =
            new Door(r1, r2);
  aMaze->AddRoom(r1);
  aMaze->AddRoom(r2);
  r1->SetSide(North,
                  new Wall);
  r1->SetSide(East, theDoor);
  r1->SetSide(South,
                  new Wall);
  r1->SetSide(West,
                  new Wall);
  r2->SetSide(North,
                  new Wall);
  r2->SetSide(East,
                  new Wall);
  r2->SetSide(South,
                  new Wall);
  r2->SetSide(West, theDoor);
  return aMaze;
}
```

Revenind la modelul în discuție, construcția labirintului va fi efectuată de către o funcție `CreateMaze` folosind un *Builder* de tip `MazeBuilder`.



Clasa `MazeBuilder` definește interfața pentru construcția labirintului:

```
class MazeBuilder {
public:
  virtual void BuildMaze(){ }
  virtual void BuildRoom(
                  int room){ }
  virtual void BuildDoor(
                  int roomFrom,
                  int roomTo){ }
  virtual Maze* GetMaze(){
    return 0;
  }

protected:
  MazeBuilder();
};
```

Această interfață poate crea trei elemente: labirintul (`Maze`), camere cu un anumit număr și uși între încăperi identificate prin numere. Metoda `GetMaze` furnizează labirintul construit.

Subclasele clasei `MazeBuilder` vor implementa această metodă pentru a furniza obiectul creat de ele.

Toate metodele de construcție ale clasei `MazeBuilder` sunt vide. Ele nu sunt declarate virtuale pure pentru a permite claselor derivate să implementeze doar metodele care sunt interesante pentru ele.

Folosind interfața `MazeBuilder`, funcția `CreateMaze` ar putea fi următoarea:

```
Maze* CreateMaze
      (MazeBuilder& builder){
  builder.BuildMaze();
  builder.BuildRoom(1);
  builder.BuildRoom(2);
  builder.BuildDoor(1, 2);

  return builder.GetMaze();
}
```

După cum se poate observa, *Builder*-ul ascunde reprezentarea internă a labirintului (clasele care definesc încăperile, pereții și ușile) precum și modul în care aceste părți sunt asamblate pentru a crea labirintul final.

Cineva ar putea ghici că există clase pentru încăperi și uși, dar nu există indicii despre pereți.

Aceasta ușurează modificarea modului în care labirintul este reprezentat, deoarece nici un client al clasei `MazeBuilder` nu trebuie modificat.

Clasa `StandardMazeBuilder` reprezintă o implementare care construiește labirinturi simple.

Ea păstrează labirintul care se construiește în variabila `currentMaze`.

```
class StandardMazeBuilder :
        public MazeBuilder {
public:
  StandardMazeBuilder();

  virtual void BuildMaze();
  virtual void
            BuildRoom(int);
  virtual void BuildDoor(int,
                  int);
  virtual Maze* GetMaze();

private:
  Direction CommonWall(Room*,
                  Room*);
  Maze* currentMaze;
};
```

Singura operație realizată de constructor este inițializarea variabilei membre:

```
StandardMazeBuilder::
      StandardMazeBuilder (){
  currentMaze = 0;
}
```

Metoda BuildMaze inițializează un obiect Maze pe care alte metode îl vor asambla sau îl vor furniza clienților.

```
void StandardMazeBuilder::
                BuildMaze () {
  currentMaze = new Maze();
}
```

Metoda GetMaze furnizează clienților interesați labirintul creat.

```
Maze* StandardMazeBuilder::
                GetMaze () {
  return currentMaze;
}
```

Metoda BuildRoom creează o încăpere și pereții din jurul ei:

```
void StandardMazeBuilder::
          BuildRoom (int n) {
  if (!currentMaze
                ->RoomNo(n)){
    Room* room = new Room(n);
    currentMaze
              ->AddRoom(room);
    room->SetSide(North, new
                    Wall());
    room->SetSide(South, new
                    Wall());
    room->SetSide(East, new
                    Wall());
    room->SetSide(West, new
                    Wall());
  }
}
```

Pentru a construi o ușă între două încăperi, StandardMazeBuilder caută ambele încăperi și peretele dintre ele:

```
void StandardMazeBuilder::
    BuildDoor(int n1, int n2){
  Room* r1 =
     currentMaze->RoomNo(n1);
  Room* r2 =
     currentMaze->RoomNo(n2);
  Door* d = new Door(r1, r2);
  r1->SetSide(
       CommonWall(r1,r2), d);
  r2->SetSide(
       CommonWall(r2,r1), d);
}
```

Secvența completă pentru crearea labirintului este:

```
Maze* maze;
StandardMazeBuilder builder;
CreateMaze(builder);
maze = builder.GetMaze();
```

O altă abordare ar fi constat în plasarea întregului cod din StandardMazeBuilder în clasa Maze și acordarea labirinturilor a "permisiunii" de a se se construi singure.

Totuși, clasa Maze, fiind mai scurtă, este mai ușor de înțeles și modificat, iar clasa StandardMazeBuilder este ușor de separat de clasa Maze.

Mai important, separarea celor două clase ne permite să creăm o varietate de clase MazeBuilder, fiecare folosind clase diferite pentru pereți, încăperi și uși.

Mai interesantă este clasa CountingMazeBuilder. Această clasă nu construiește un labirint, ci doar numără cantitățile folosite din diferite componente (camera și uși) pentru a crea un labirint.

```
class CountingMazeBuilder :
          public MazeBuilder {
public:
  CountingMazeBuilder();

  virtual void BuildMaze();
  virtual void
                BuildRoom(int);
  virtual void BuildDoor(int,
                         int);
  void GetCounts(int&, int&)
                         const;
private:
  int doors;
  int rooms;
};
```

Constructorul inițializează cele două contoare, iar metodele supraîncărcate le incrementează corespunzător (pentru momentul în care elementele sunt utilizate):

```
CountingMazeBuilder::
     CountingMazeBuilder () {
  rooms = 0;
  doors = 0;
}

void CountingMazeBuilder::
          BuildRoom (int) {
  rooms++;
}

void CountingMazeBuilder::
        BuildDoor (int, int) {
  doors++;
}

void CountingMazeBuilder::
   GetCounts (int& r, int& d)
                      const {
  r = rooms;
  d = doors;
}
```

Iată cum ar putea fi folosită această clasă:

```
int rooms, doors;
CountingMazeBuilder builder;
CreateMaze(builder);
builder.GetCounts(rooms,
                      doors);
cout << "The maze has "
       << rooms
       << " rooms and "
       << doors
       << " doors"
       << endl;
```

Va urma...

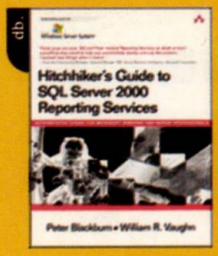
Sperăm că prin acest model și exemplificarea lui v-am stârnit interesul pentru design în general și modele de design în particular. Vom continua în numerele următoare prezentarea celor mai cunoscute modele de proiectare.

Ce mai e nou?

coming soon

Cisco IP Telephony Flash Cards and Exam Practice Pack
By Kevin Wallace / Oct 7, 2004

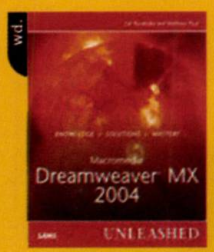
Design and Deploy Websites with Macromedia Dreamweaver MX 2004
By Joseph Lowery / Oct 29, 2004



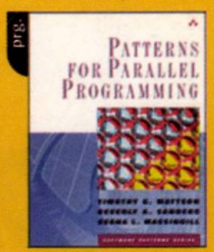
Hitchhiker's Guide to SQL Server 2000 Reporting Services
By Peter Blackburn, William Vaughn
Oct 8, 2004

Create Your Own Website (Using What you Already Know)
By Scott Mitchel / Oct 7, 2004

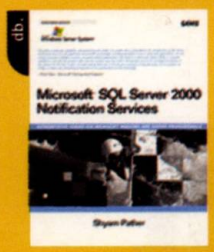
just released



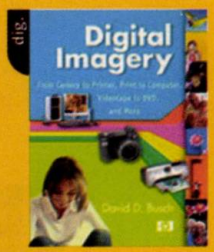
Macromedia Dreamweaver MX 2004 Unleashed
By Zak Ruvalcaba / Sep 21, 2004



Patterns for Parallel Programming
By Timothy Mattson / Sep 15, 2004

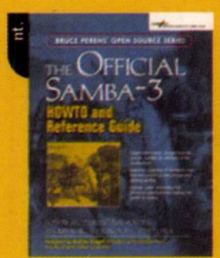


Microsoft SQL Server 2000 Notification Services
By Shyam Pather / Sep 21, 2004



Digital Photography: From Camera to Printer, Print to Computer
By David Busch / Sep 23, 2004

most popular



Official Samba-3 HOWTO and Reference Guide, The
By John Terpstra, / Oct 22, 2003

C++ Without Fear: A Beginner's Guide That Makes You Feel Smart
By Brian Overland / Sep 14, 2004

High-Tech Crimes Revealed: Cyberwar Stories from the Digital Front
By Steven Branigan / Aug 27, 2004

Database Programming with Visual Basic .NET and ADO.NET
By F. Scott Barker / Sep 12, 2002

Fii pe fază!

Cele mai noi, cele mai bune cărți de IT&C

PEARSON Education

BYBLOS

Byblos srl, tel: 021-330.92.81, fax: 021-330.92.85, e-mail: dpastrama@byblos.ro
www.byblos.ro
Librăria Eminescu, Universitate, la subsol, tel: 021-312.72.65, 0721-927.499

Publicarea acestei reviste este posibilă datorită companiilor de mai jos care sprijină învățământul românesc de informatică

SPONSORI DE PLATINĂ

FORTE COMPUTERS
Flexibility. Made in Romania

IBM®

s&t
IT SOLUTIONS & SERVICES

hp®

SPONSOR DE AUR

CCT
CABINETUL CĂLIN TATOMIR

Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-266.516